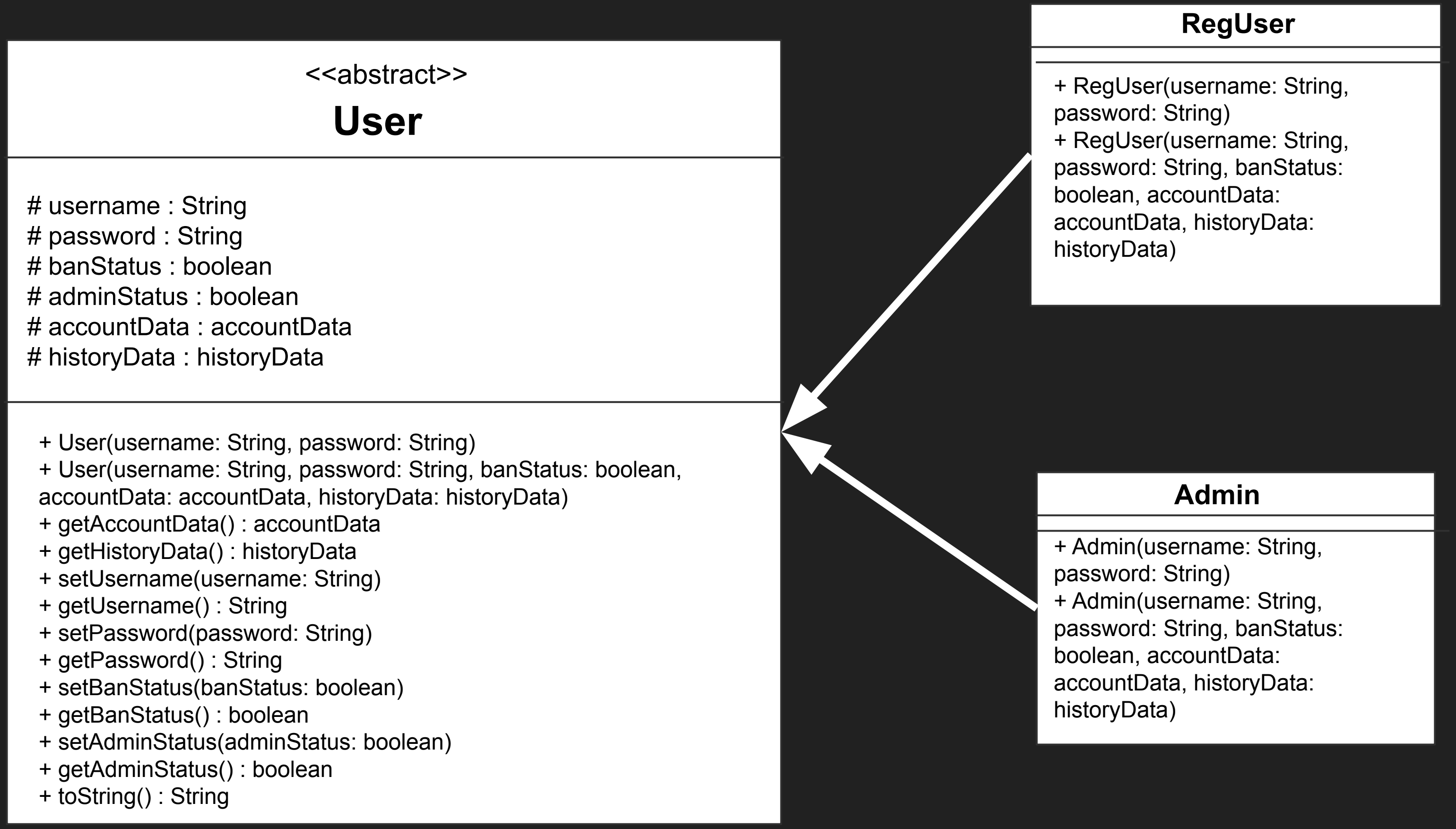
design1

LOGIN/USER SYSTEM

CURRENT DESIGN
(for phase1)

Note: a better design for
User is planned for phase2



<<abstract>>
User

# username : String
# password : String
# banStatus : boolean
# adminStatus : boolean
# accountData : accountData
# historyData : historyData

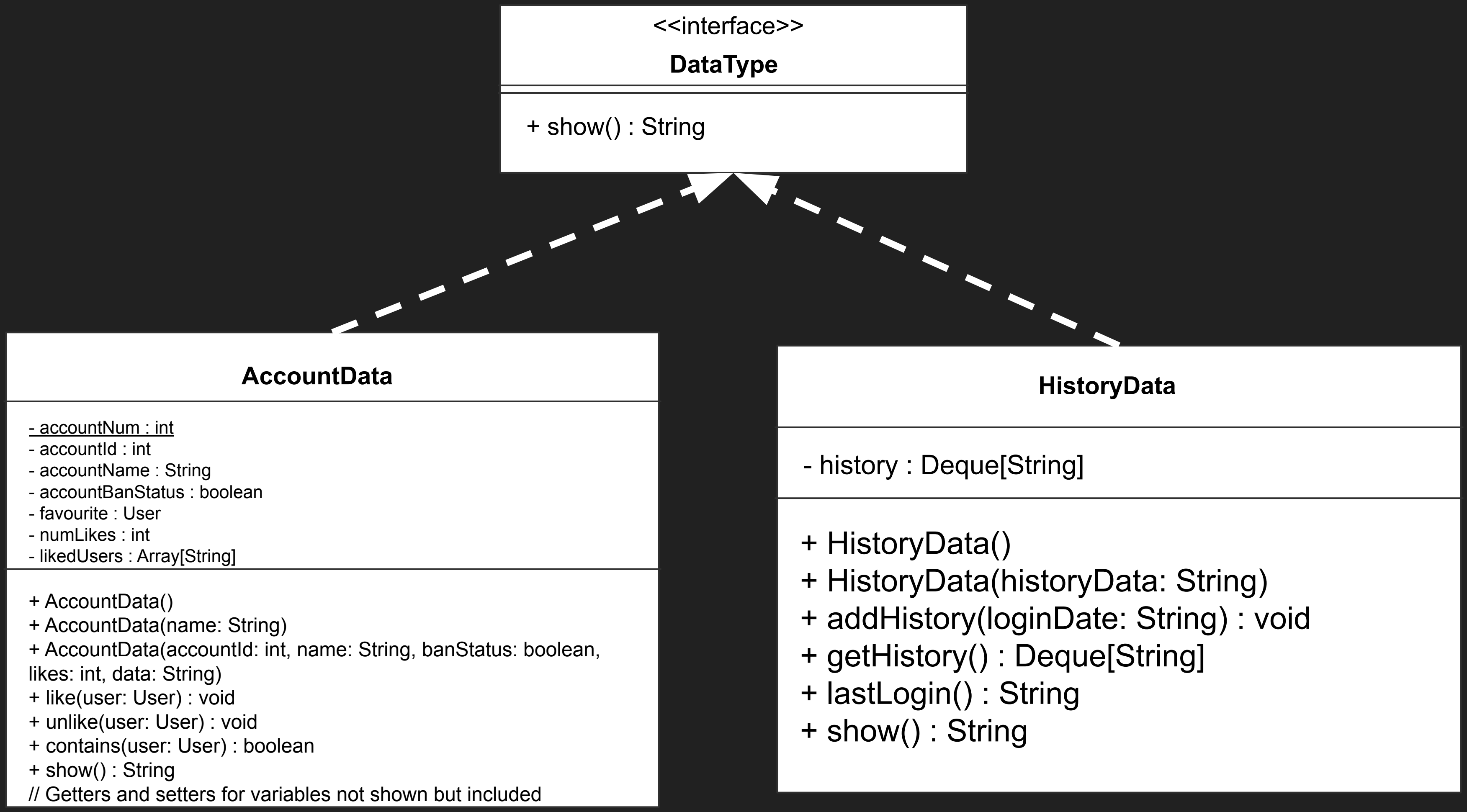
+ User(username: String, password: String)
+ User(username: String, password: String, banStatus: boolean, accountData: accountData, historyData: historyData)
+ getAccountData() : accountData
+ getHistoryData() : historyData
+ setUsername(username: String)
+ getUsername() : String
+ setPassword(password: String)
+ getPassword() : String
+ setBanStatus(banStatus: boolean)
+ getBanStatus() : boolean
+ setAdminStatus(adminStatus: boolean)
+ getAdminStatus() : boolean
+ toString() : String

RegUser

+ RegUser(username: String, password: String)
+ RegUser(username: String, password: String, banStatus: boolean, accountData: accountData, historyData: historyData)

Admin

+ Admin(username: String, password: String)
+ Admin(username: String, password: String, banStatus: boolean, accountData: accountData, historyData: historyData)



<<interface>>
DataType

+ show() : String

AccountData

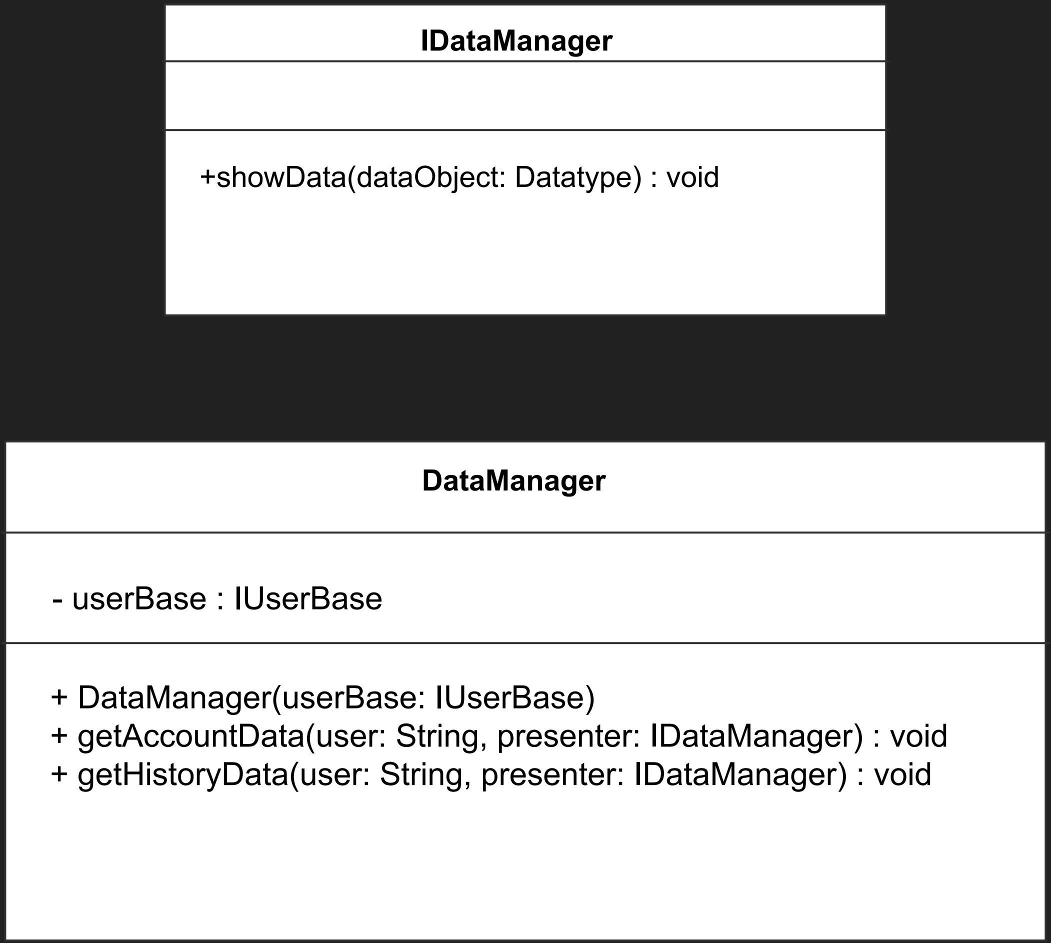
- accountNum : int
- accountId : int
- accountName : String
- accountBanStatus : boolean
- favourite : User
- numLikes : int
- likedUsers : Array[String]

+ AccountData()
+ AccountData(name: String)
+ AccountData(accountId: int, name: String, banStatus: boolean, likes: int, data: String)
+ like(user: User) : void
+ unlike(user: User) : void
+ contains(user: User) : boolean
+ show() : String
// Getters and setters for variables not shown but included

HistoryData

- history : Deque[String]

+ HistoryData()
+ HistoryData(historyData: String)
+ addHistory(loginDate: String) : void
+ getHistory() : Deque[String]
+ lastLogin() : String
+ show() : String



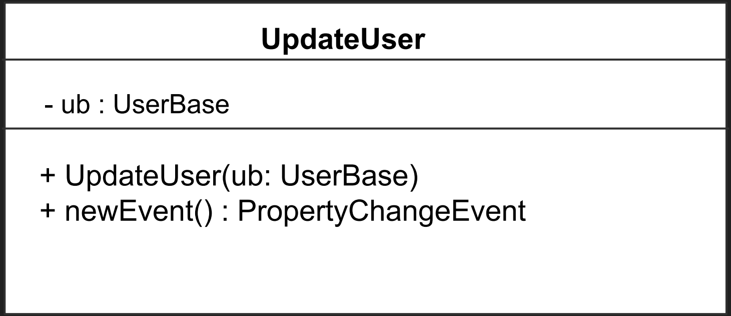
IDataManager

+showData(dataObject: Datatype) : void

DataManager

- userBase : IUserBase

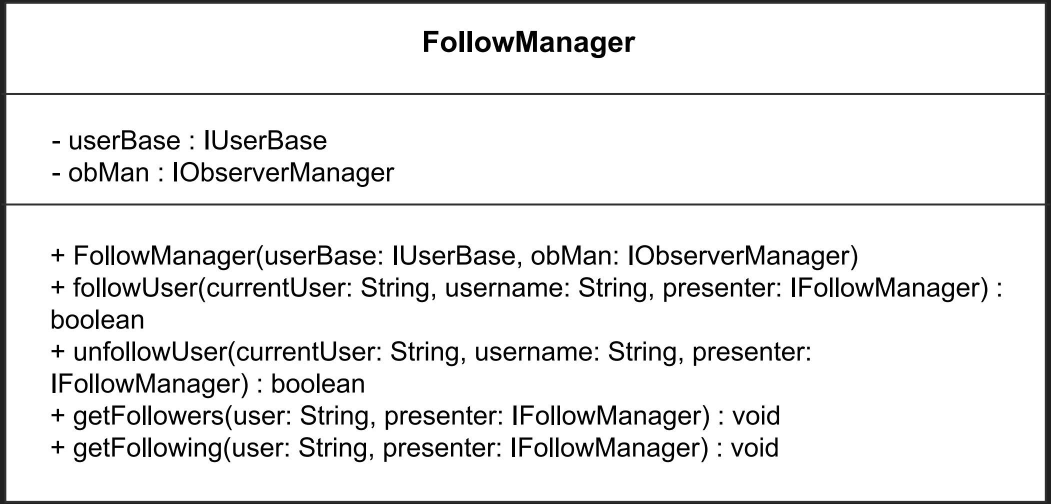
+ DataManager(userBase: IUserBase)
+ getAccountData(user: String, presenter: IDataManager) : void
+ getHistoryData(user: String, presenter: IDataManager) : void



UpdateUser

- ub : UserBase

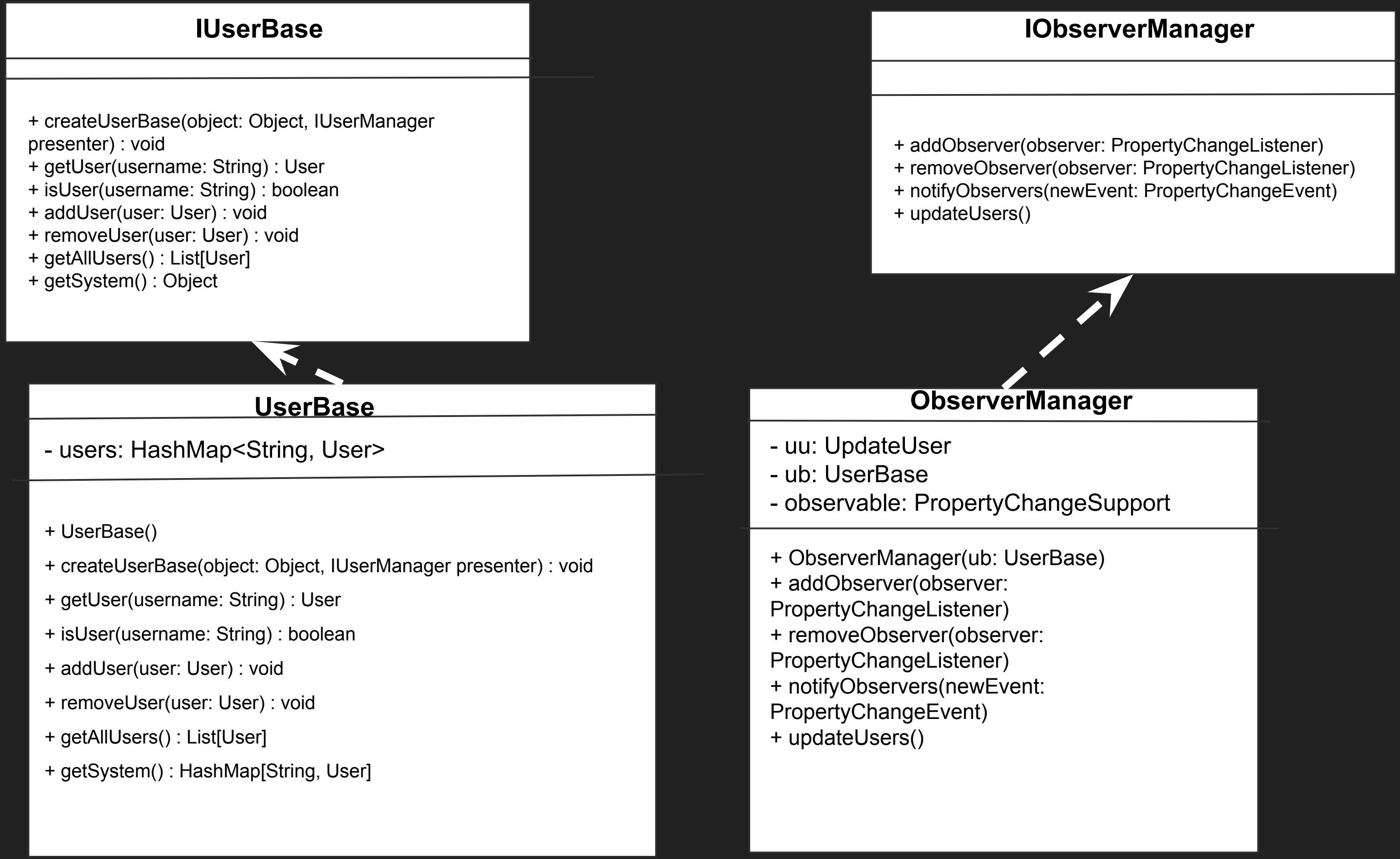
+ UpdateUser(ub: UserBase)
+ newEvent() : PropertyChangeEvent



FollowManager

- userBase : IUserBase
- obMan : IObserverManager

+ FollowManager(userBase: IUserBase, obMan: IObserverManager)
+ followUser(currentUser: String, username: String, presenter: IFollowManager) : boolean
+ unfollowUser(currentUser: String, username: String, presenter: IFollowManager) : boolean
+ getFollowers(user: String, presenter: IFollowManager) : void
+ getFollowing(user: String, presenter: IFollowManager) : void



IUserBase

+ createUserBase(object: Object, IUserManager presenter) : void
+ getUser(username: String) : User
+ isUser(username: String) : boolean
+ addUser(user: User) : void
+ removeUser(user: User) : void
+ getAllUsers() : List[User]
+ getSystem() : Object

IObserverManager

+ addObserver(observer: PropertyChangeListener)
+ removeObserver(observer: PropertyChangeListener)
+ notifyObservers(newEvent: PropertyChangeEvent)
+ updateUsers()

UserBase

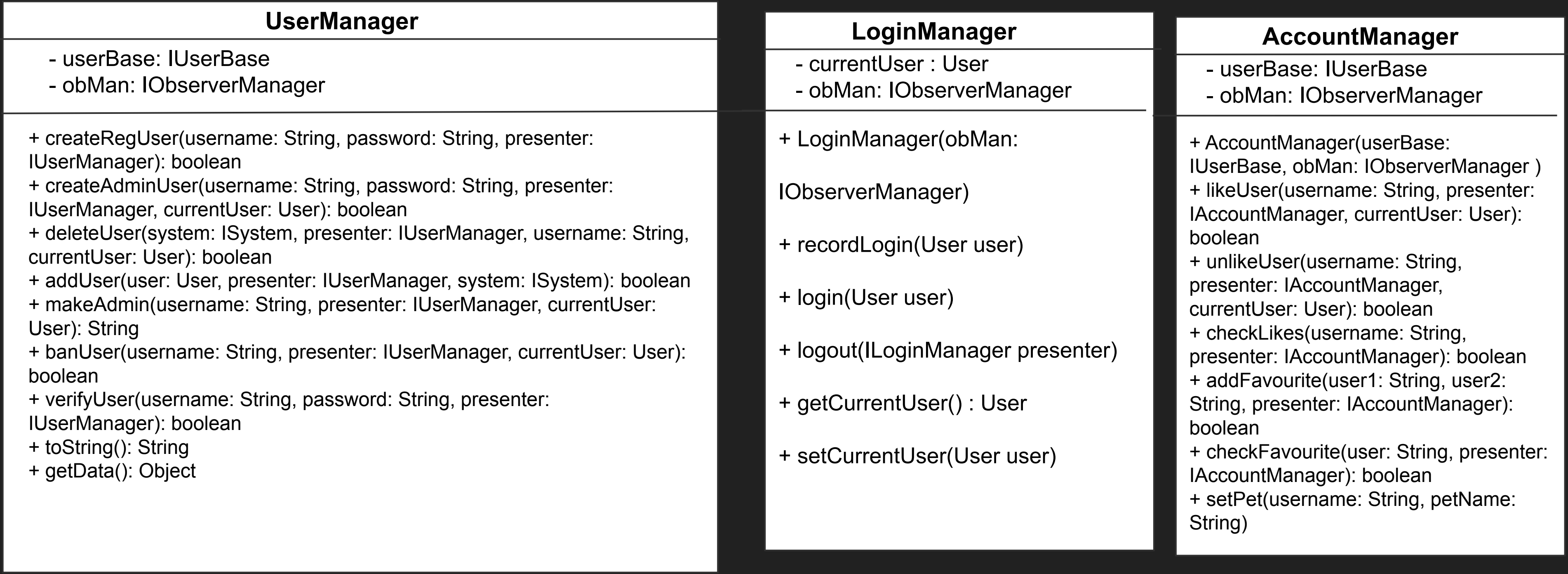
- users: HashMap<String, User>

+ UserBase()
+ createUserBase(object: Object, IUserManager presenter) : void
+ getUser(username: String) : User
+ isUser(username: String) : boolean
+ addUser(user: User) : void
+ removeUser(user: User) : void
+ getAllUsers() : List[User]
+ getSystem() : HashMap[String, User]

ObserverManager

- uu: UpdateUser
- ub: UserBase
- observable: PropertyChangeSupport

+ ObserverManager(ub: UserBase)
+ addObserver(observer: PropertyChangeListener)
+ removeObserver(observer: PropertyChangeListener)
+ notifyObservers(newEvent: PropertyChangeEvent)
+ updateUsers()



UserManager

- userBase: IUserBase
- obMan: IObserverManager

+ createRegUser(username: String, password: String, presenter: IUserManager): boolean
+ createAdminUser(username: String, password: String, presenter: IUserManager, currentUser: User): boolean
+ deleteUser(system: ISystem, presenter: IUserManager, username: String, currentUser: User): boolean
+ addUser(user: User, presenter: IUserManager, system: ISystem): boolean
+ makeAdmin(username: String, presenter: IUserManager, currentUser: User): String
+ banUser(username: String, presenter: IUserManager, currentUser: User): boolean
+ verifyUser(username: String, password: String, presenter: IUserManager): boolean
+ toString(): String
+ getData(): Object

LoginManager

- currentUser : User
- obMan: IObserverManager

+ LoginManager(obMan:

IObserverManager)

+ recordLogin(User user)

+ login(User user)

+ logout(ILoginManager presenter)

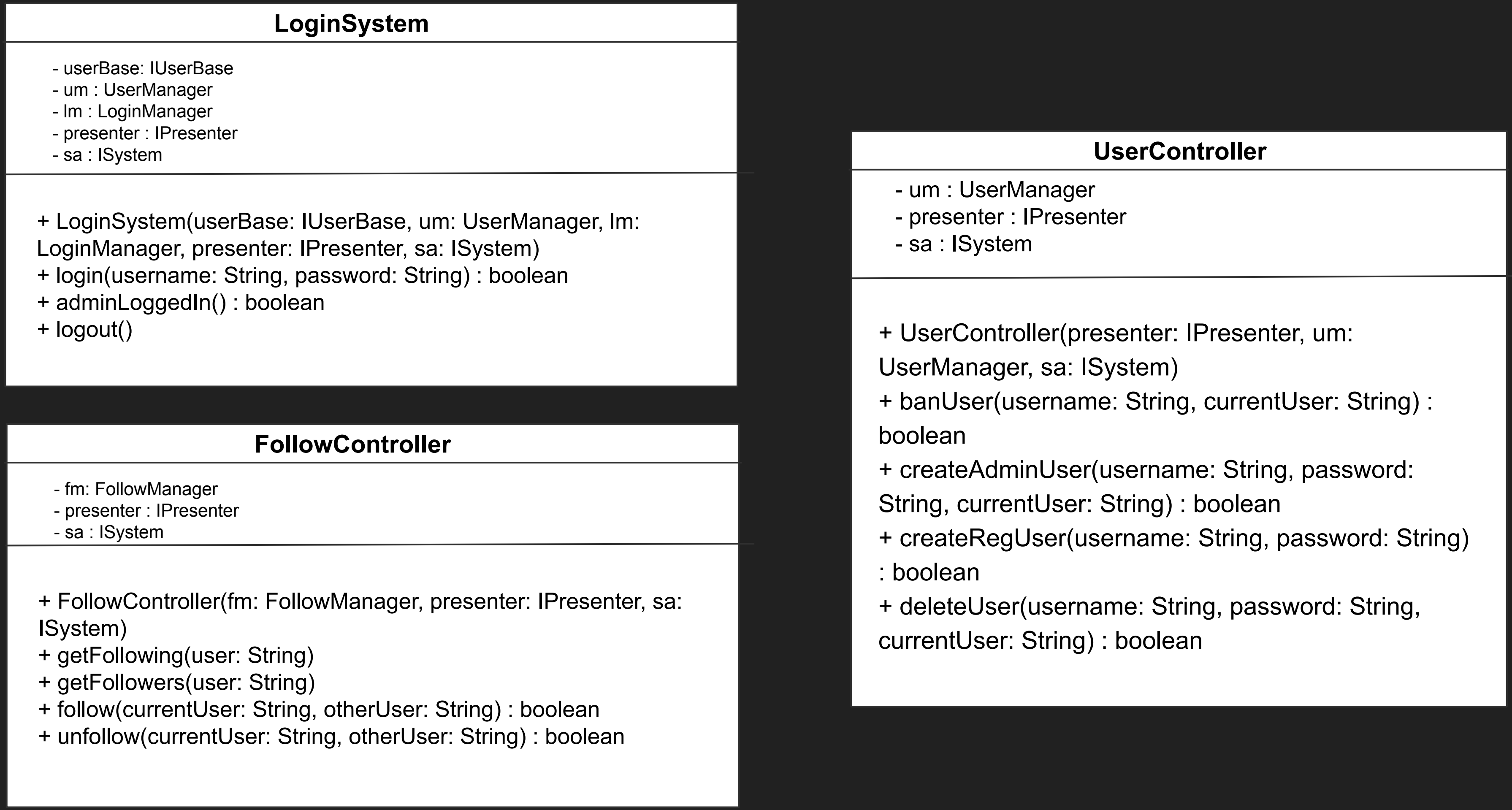
+ getCurrentUser() : User

+ setCurrentUser(User user)

AccountManager

- userBase: IUserBase
- obMan: IObserverManager

+ AccountManager(userBase: IUserBase, obMan: IObserverManager)
+ likeUser(username: String, presenter: IAccountManager, currentUser: User): boolean
+ unlikeUser(username: String, presenter: IAccountManager, currentUser: User): boolean
+ checkLikes(username: String, presenter: IAccountManager): boolean
+ addFavourite(user1: String, user2: String, presenter: IAccountManager): boolean
+ checkFavourite(user: String, presenter: IAccountManager): boolean
+ setPet(username: String, petName: String)



LoginSystem

- userBase: IUserBase
- um : UserManager
- lm : LoginManager
- presenter : IPresenter
- sa : ISystem

+ LoginSystem(userBase: IUserBase, um: UserManager, lm: LoginManager, presenter: IPresenter, sa: ISystem)
+ login(username: String, password: String) : boolean
+ adminLoggedIn() : boolean
+ logout()

FollowController

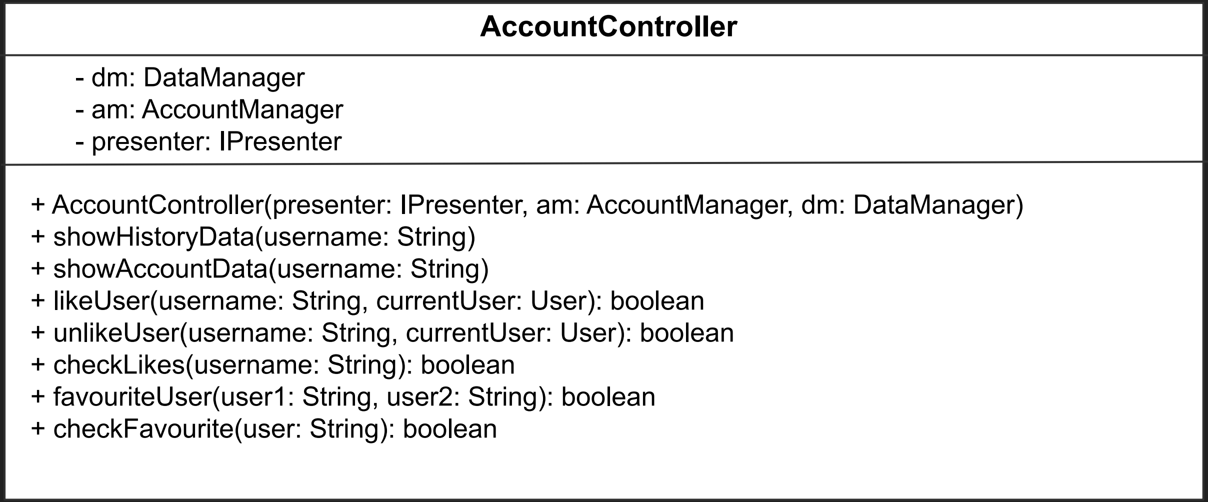
- fm: FollowManager
- presenter : IPresenter
- sa : ISystem

+ FollowController(fm: FollowManager, presenter: IPresenter, sa: ISystem)
+ getFollowing(user: String)
+ getFollowers(user: String)
+ follow(currentUser: String, otherUser: String) : boolean
+ unfollow(currentUser: String, otherUser: String) : boolean

UserController

- um : UserManager
- presenter : IPresenter
- sa : ISystem

+ UserController(presenter: IPresenter, um: UserManager, sa: ISystem)
+ banUser(username: String, currentUser: String) : boolean
+ createAdminUser(username: String, password: String, currentUser: String) : boolean
+ createRegUser(username: String, password: String) : boolean
+ deleteUser(username: String, password: String, currentUser: String) : boolean

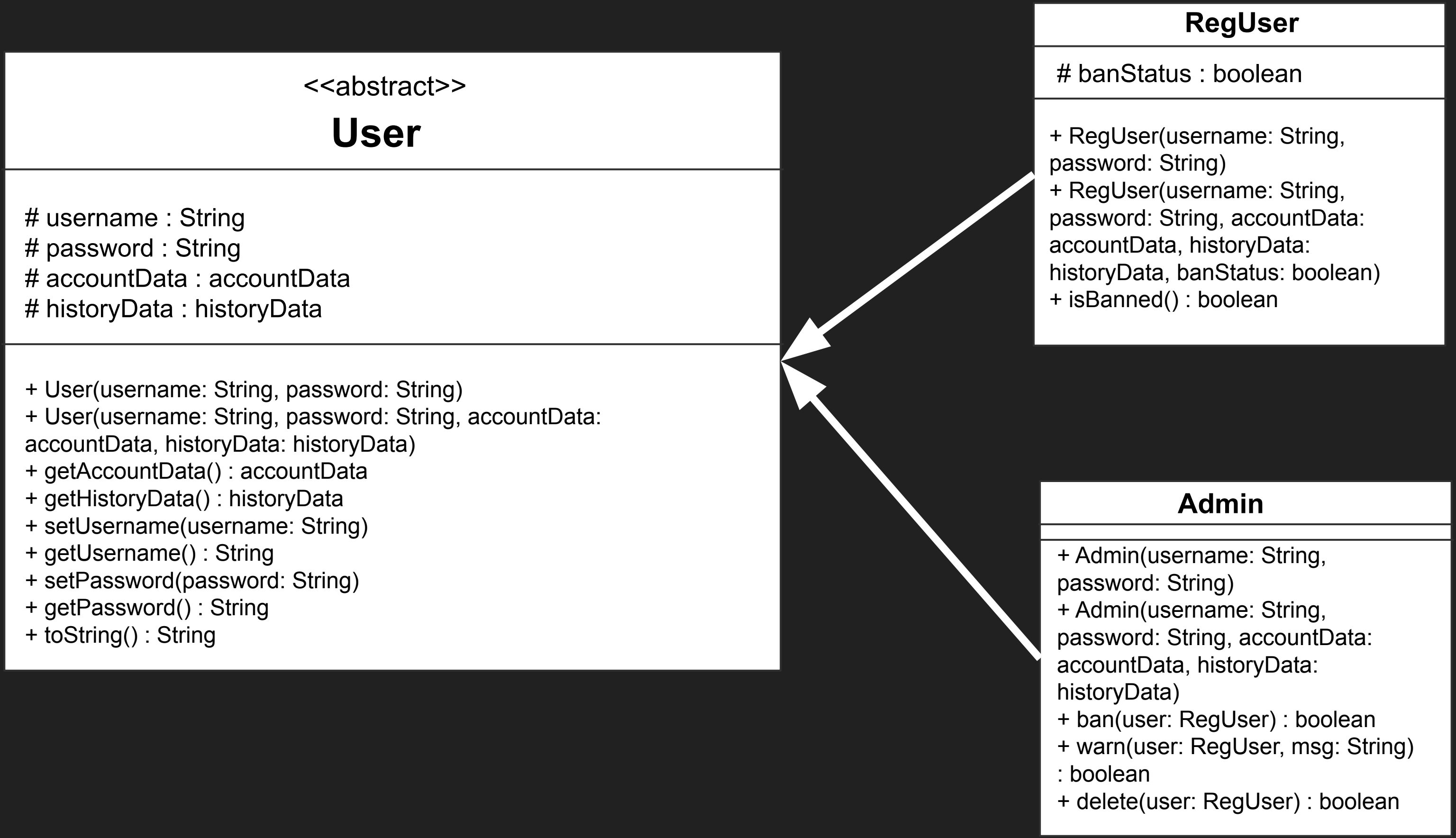


AccountController

- dm: DataManager
- am: AccountManager
- presenter: IPresenter

+ AccountController(presenter: IPresenter, am: AccountManager, dm: DataManager)
+ showHistoryData(username: String)
+ showAccountData(username: String)
+ likeUser(username: String, currentUser: User): boolean
+ unlikeUser(username: String, currentUser: User): boolean
+ checkLikes(username: String): boolean
+ favouriteUser(user1: String, user2: String): boolean
+ checkFavourite(user: String): boolean

NEW DESIGN
(for phase2)



<<abstract>>

User

# username : String
# password : String
# accountData : accountData
# historyData : historyData

+ User(username: String, password: String)
+ User(username: String, password: String, accountData: accountData, historyData: historyData)
+ getAccountData() : accountData
+ getHistoryData() : historyData
+ setUsername(username: String)
+ getUsername() : String
+ setPassword(password: String)
+ getPassword() : String
+ toString() : String

RegUser

# banStatus : boolean

+ RegUser(username: String, password: String)
+ RegUser(username: String, password: String, accountData: accountData, historyData: historyData, banStatus: boolean)
+ isBanned() : boolean

Admin

+ Admin(username: String, password: String)
+ Admin(username: String, password: String, accountData: accountData, historyData: historyData)
+ ban(user: RegUser) : boolean
+ warn(user: RegUser, msg: String) : boolean
+ delete(user: RegUser) : boolean

OTHER PLANNED UPGRADES (for phase2)

AdminManager and AdminController

Manages all admin functionality

- Distinct from UserManager
- Single responsibility principle
- Reduces bloated code smell
- Can incorporate a facade design pattern…