

UML for Use Cases

userManager
-listTradableUser: ArrayList<TradableUser> -listAdmin: ArrayList<User> -uCommunityM: UserCommunityManager -uInfoM: UserInfoManager -uItemM: UserItemManager -uThresholdM: UserThresholdManager
+userManager() +userManager(tradableUsers: ArrayList<TradableUser>, admins: ArrayList<User>) +getListTradableUser(): ArrayList<TradableUser> +getListUnfreezeRequest(): ArrayList<String[]> +freezeUser(username: String): boolean +unfreezeUser(username: String): boolean +checkUser(username: String): boolean +checkUser(userID: int): boolean +addUser(username: String, password: String, email: String, home: String): void +addAdmin(username: String, password: String, email: String): void +removeItemWishlist(itemID: Integer, username: String): boolean +removeItemInventory(itemID: Integer, username: String): boolean +addItemWishlist(itemID: Integer, username: String): boolean +addItemInventory(itemID: Integer, username: String): boolean +userPasswords(): HashMap<String, String> +adminPasswords(): HashMap<String, String> +findUser(username: String): TradableUser +findUser(ID: int): TradableUser +idToUsername(id: int): String +usernameToID(username: String): int +requestUnfreeze(username: String, message: String): boolean +removeItemFromUsers(userID1: int, userID2: int, itemID: int): void +getFrozenStatus(username: String): boolean +getFrozenStatus(userID: int): boolean +getUserInventory(userID: int): ArrayList<Integer> +getUserWishlist(userID: int): ArrayList<Integer> +setThreshold(userID: int, threshold: String, change: int): void +getInfo(userID: int, threshold: String): int +getInfo(username: int, threshold: String): int +getFriends(userID: int): ArrayList<TradableUser> +requestFriend(message: String, userTo: String, userFrom: String): boolean +addFriend(user1: String, user2: String): boolean +removeFriend(user1: String, user2: String): boolean +removeFriend(user1: int, user2: int): boolean +goOnVacation(userID: int): boolean +comeFromVacation(userID: int): boolean +sameCity(userID: int): ArrayList<TradableUser> +wantedItems(wantUser: int, haveUser: int): ArrayList<Integer> +getHome(userID: int): String +changeHome(userID: int, newHome: String): void +userFollow(userID: int, toFollow: int): boolean +userUnfollow(userID: int, toUnfollow: int): boolean +itemFollow(userID: int, toFollow: Item): boolean +itemUnfollow(userID: int, toUnfollow: Item): boolean +itemsFollowed(): HashMap<Integer, ArrayList<Integer>> +friendsRequesting(userID: int): ArrayList<String[]> +getUserFollowingLogs(userID: int): ArrayList<String> +getItemFollowingLogs(userID: int): ArrayList<String> +sortRating(fm: FeedbackManager): ArrayList<TradableUser> +getUsersNotFriends(userID: int): ArrayList<TradableUser>

<<interface>> Serializable
UserInfoManager
#checkUser(username: String, listTradableUser: ArrayList<TradableUser>): boolean #checkUser(userID: int, listTradableUser: ArrayList<TradableUser>): boolean #userPasswords(listTradableUser: ArrayList<TradableUser>): HashMap<String, String> #adminPasswords(listAdmin: ArrayList<User>): HashMap<String, String> #findUser(username: String, listTradableUser: ArrayList<TradableUser>): TradableUser #findUser(ID: int, listTradableUser: ArrayList<TradableUser>): TradableUser #idToUsername(ID: int, listTradableUser: ArrayList<TradableUser>, listAdmin: ArrayList<User>): String #usernameToID(username: String, listTradableUser: ArrayList<TradableUser>, listAdmin: ArrayList<User>): int #getHome(userID: int, listTradableUser: ArrayList<TradableUser>): String #changeHome(userID: int, newHome: String, listTradableUser: ArrayList<TradableUser>): void
UserItemManager
#removeItemWishlist(itemID: Integer, username: String, person: TradableUser, uim: UserInfoManager, ucm: UserCommunityManager, listTradableUser: ArrayList<TradableUser>, listAdmin: ArrayList<User>): boolean #removeItemInventory(itemID: Integer, username: String, person: TradableUser, uim: UserInfoManager, ucm: UserCommunityManager, listTradableUser: ArrayList<TradableUser>, listAdmin: ArrayList<User>): boolean #addItemWishlist(itemID: Integer, username: String, person: TradableUser, uim: UserInfoManager, ucm: UserCommunityManager, listTradableUser: ArrayList<TradableUser>, listAdmin: ArrayList<User>): boolean #addItemInventory(itemID: Integer, username: String, person: TradableUser, uim: UserInfoManager, ucm: UserCommunityManager, listTradableUser: ArrayList<TradableUser>, listAdmin: ArrayList<User>): boolean #removeItemFromUsers(itemID: int, user1: TradableUser, user2: TradableUser, uim: UserInfoManager, ucm: UserCommunityManager, listTradableUser: ArrayList<TradableUser>, listAdmin: ArrayList<User>): void #getUserInventory(person: TradableUser): ArrayList<Integer> #getUserWishlist(person: TradableUser): ArrayList<Integer> #wantedItems(person1: TradableUser, person2: TradableUser): ArrayList<Integer>

UserCommunityManager
-listFriendRequest: ArrayList<String[]>
+UserCommunityManager() #usersFollowingUser(userID: int, person: TradableUser, listTradableUser: ArrayList<TradableUser>): ArrayList<TradableUser> #editFollowerLogs(toAdd: String, userID: int, person: TradableUser, listTradableUser: ArrayList<TradableUser>): void #userFollow(userID: int, toFollow: int, person: TradableUser, following: TradableUser, listTradableUser: ArrayList<TradableUser>): boolean #userUnfollow(userID: int, toUnfollow: int, person: TradableUser, following: TradableUser, listTradableUser: ArrayList<TradableUser>): boolean #itemFollow(userID: int, toFollow: Item, person: TradableUser, listTradableUser: ArrayList<TradableUser>): boolean #getUserFollowingLogs(person: TradableUser): ArrayList<String> #getItemFollowingLogs(person: TradableUser): ArrayList<String> #itemsFollowed(listTradableUser: ArrayList<TradableUser>): HashMap<Integer, ArrayList<Integer>> #getFriends(person: TradableUser, uim: UserInfoManager, listTradableUser: ArrayList<TradableUser>): ArrayList<TradableUser> #requestFriend(message: String, userTo: String, userFrom: String): boolean #addFriend(user1: int, user2: int, person1: TradableUser, person2: TradableUser, listTradableUser: ArrayList<TradableUser>, uim: UserInfoManager, listAdmin: ArrayList<TradableUser>): boolean #removeFriend(user1: int, user2: int, person1: TradableUser, person2: TradableUser): boolean #friendsRequesting(username: String): ArrayList<String[]> #getUsersNotFriends(userID: int, uim: UserInfoManager, username: String, listTradableUser: ArrayList<TradableUser>, listAdmin: ArrayList<User>): ArrayList<TradableUser> #sameCity(homosapien: TradableUser, listTradableUser: ArrayList<TradableUser>): ArrayList<TradableUser> -merge(list1: ArrayList<TradableUser>, list2: ArrayList<TradableUser>, fm: FeedbackManager): ArrayList<TradableUser> -mergeSort(list: ArrayList<TradableUser>, fm: FeedbackManager): ArrayList<TradableUser> #sortRating(fm: FeedbackManager, listTradableUser: ArrayList<TradableUser>): ArrayList<TradableUser>
UserThresholdManager
-listUnfreezeRequest: ArrayList<String[]>
+UserThresholdManager() #freezeUser(person: TradableUser): boolean #unfreezeUser(username: String, person: TradableUser): boolean #requestUnfreeze(username: String, message: String, person: TradableUser): boolean #getFrozenStatus(person: TradableUser): boolean #setThreshold(threshold: String, change: int, person: TradableUser): void #getInfo(threshold: String, person: TradableUser): int #getListUnfreezeRequest(): ArrayList<String[]> #goOnVacation(userID: int, person: TradableUser, ucm: UserCommunityManager, listTradableUser: ArrayList<TradableUser>): boolean #comeFromVacation(userID: int, person: TradableUser, ucm: UserCommunityManager, listTradableUser: ArrayList<TradableUser>): boolean