Bryan Mundia, Eric Park, Sean Htet, Martin Tran, & Cyrus Foroudian
Professor Abolghasemi
Final Project UML Diagram

FinalProject - Facebook 2.0

ProfileManager

- profiles for UndirectedGraph
- sc for Scanner.

+ **addProfile**(self): creates a new profile.
a. calls add() from CustomHashSet to add user profile.
b. Print statement for when profile has been added successfully.

+ **removeProfile**(): method to remove the user profile:
a. calls remove() from CustomHashSet to remove user profile.
b. Print statement for when profile has been removed successfully.

+ **modifyProfile**(): method to let user to modify their profile further:
a. Change online status.
b. Change name or include a nickname.
c. Change user picture.
d. Add or remove friends.
e. Add personal information.
f. Disable / reactivate account.
g. Search for othe rprofiles.
h. Show friends.
i. Exit from modifying.

+ **modifyFriends**(): method to add or remove people from their friendlist:
a. Switch statements that lets a user to remove, add, or exit modifying their friendlist.

+ **searchProfiles**(): method to searches a persons profile:
a. calls getProfileByName and prints if profile found or not.

+ **showFriends**(): method to show users friend list:
a. calls getFriends() and prints out friend list.

- **getProfileByName**(): method to get persons profile name:
a. Compares from the user input that if a name is in the user profile database.

- **findFriend(**Profile profile, String friendName**): Profile**
This method retrieves a list of a profile's friend' friends.

- **addFriend**(): method to add a friend to the users friends list:
a. loop with conditional statements that checks if user input is in the profile database, then adds the person.

- **removeFriend**(): method to remove a friend from the users friends list:
a. Conditional statement that friend was removed successfully.

- **addPersonalInformation**(): method add users music, movie, and hobbies:
a. Conditional statements that check each 3 additional info are added correctly.

- **toggleAccountStatus**(): method to re-active or disable account:
a. calls setActive & isActive to confirm if account is reactived or disabled.

ProjectDriver (main)

No attributes. The main driver of the entire project.

Objects created to call out the ProfileManager & Profile to run.
- Has a loop to add in further information after user profile has been created.

Profile

- name: String
- profileImage: File
- age: int
- status: String
- statusMessage: String
- sc: Scanner
- isActive: boolean
- music: String
- movie: String
- hobby: String
- friends: Set<Profile>
- getFileExtension(File file): String
- isSupportedImage(String extension): boolean

+ Profile()
+ Profile(name: String, age: int, status: String, statusMessage: String, profileImage: File)
+ setStatus()
+ setName()
+setAge()
+ setStatusMessage()
+ setProfileImage()
+ setMusic(music: String)
+ setMovies(movie: String)
+ setHobbies(hobby: String)
+ setActive(active: boolean)
+ getName(): String
+ getAge(): int
+ getProfileImage(): File
+ getStatus(): String
+ getStatusMessage(): String
+ isActive(): boolean
+ getFriends(): Set<Profile>
+ printProfile(): String
+ equals(other: Profile): boolean
+ toString(): String

CustomHashSet

int DEFAULT_CAPACITY = 10
T[] list
int size
- empList
- size = 0.
- list = tempList

+ **getCurrentsize**(self): gets the current size of the hash set.
a. Returns size.

+ **isEmpty**(): states if the hash set is empty or not:
a. With a return statement.

+ **add**(): method to add new entry to hash set:
a. If entry was successful, then return true.

+ **remove**(): method to remove entry from the hash set:
a. If entry was successful, then return true. Otherwise, return false.

+ **clear**(): method to clear everything from the hash set:
a. Resets everything.

+ **contains**(): method to check if user entry is in hash set:
a. Return true or false.

+ **toArray**(): method to state what is in the list and size:
a. Returns the amount in the list and size.

+ **indexOf**(): method to return the index of user entry:
a. Check if there was a match, then return.

+ **ensureCapacity**(): method to check if capacity is not full, if it is then double the size:
a. conditional statements used.

+ **toString**(): method to change array to string:
a. returns the array.

+ **CustomHashSetIterator**(): inner class to construct a new custom hash set:
a. currentIndex = 0 (for the constructor).

<<interface>> SetInterface<T> & Iterator Interface

+ getCurrentSize(): int
+ isEmpty(): boolean
+ add(T): boolean
+ remove(T) : boolean
+ contains(T): boolean
+ clear(): void
+ toArray(): T[]
+ indexOf(T) : int
+ toString(): String

CustomNoSuchElementException

+ CustomNoSuchElementException()
+ CustomNoSuchElementException(String message)

CustomUnsupportedOperationException

When an unsupported operation is attempted, the class will throw an exception.
- Default has no detailed message.
- Other has a detailed message.

UndirectedGraph<T>

-friendsMap: HashMap<T, CustomHashSet<T>>
- edgeCount : int

+ UndirectedGraph()
+addVertex(T): boolean
+removeVertex(T): void
+ addEdge(T,T,double) : boolean
+addEdge(T,T): boolean
+ hasEdge(T,T): boolean
+isEmpty(): boolean
+ getNumberOfVertices(): int
+getNumberOfEdges(): int
+clear(): void
+removeEdge(T,T): boolean

<<interface>> GraphInterface<T>

+ addVertex(T): void
+ removeVertex(T):void
+addEdge(T,T, double): boolean
+addEdge(T,T): boolean
+hasEdge(T,T) : boolean
+removeEdge(T,T): boolean
+ isEmpty(): boolean
+getNumberOfEdges(): int
+getNumberOfVertices(): int
+clear():void

<<interface>> MapInterface<K,V>

+ put(K,V): V
+remove(K) : V
+get(K): V
+containsKey(K): boolean
+containsValue(V): boolean
+size(): int
+keySet(): CustomHashSet<K>
+valueSet(): CustomHashSet<V>
+entrySet():Custom<TableEntry<K,V>>
+ isEmpty(): boolean
+clear():void

TableEntry<K, V>

-K key
-V value

+ UndirectedGraph()
+getKey(): key
+getValue(): value
+ setValue(V newValue) : void

HashMap

_int numberOfEntries
_int DEFAULT_CAPACITY = 5
_int MAX_CAPACITY = 10000
_TableEntry<k,v>[] hashTable
_int tableSize
_int MAX_SIZE = 2 * MAX_CAPACITY
_boolean initialized = false
_double MAX_LOAD_FACTOR = 0.75

+ **HashMap**(self): this(DEFAULT_CAPACITY).

+ **HashMap**(int initialCapacity): Creates a new HashMap instance with the specified initial capacity.

+ **put**(key, value): Adds or updates a key-value pair in the hash map. If the hash table becomes too full, it will be enlarged.

+ **remove**(key): Removes the entry with the specified key from the hash map.

+ **get**(key): Retrieves the value associated with the specified key from the hash map.

+ **containsKey**(key): Checks whether the hash map contains the specified key.

+ **containsValue**(value): Checks whether the hash map contains the specified value.

+ **isEmpty**(): Checks if the hash map is empty.

+ **size**(): Returns the number of entries in the hash map.

+ **clear**(): Removes all entries from the hash map.

+ **getHashIndex**(key): Calculates the hash index for a given key.

- **getNextPrime**(integer): Finds the next prime number greater than the given integer.

- **isPrime**(integer): Checks if the given integer is a prime number.

- **probe**(index, key): Probes for the next available index using linear probing.

- **locate**(): Locates the index of the entry with the specified key.

- **checkInitialization**(): Checks if the hash map has been initialized.

- **checkCapacity**(capacity): Checks if the specified capacity is within the allowed range.

- **check**(size): Checks if the size of the dictionary is too large or not.

- **enlargeHashTable**(): Doubles the size of the hash table and rehashes all the entries.

- **isHashTableTooFull**(): Checks if the hash map is too full, based on the maximum load factor.

- **valueSet**(): Sets the value.

- **entrySet**(): Sets the entry.