

AbstractSet<Ma

Entr

Exte

+ size() : int + iterator() : Iterator<

p.Entry<K, V>>

l ends

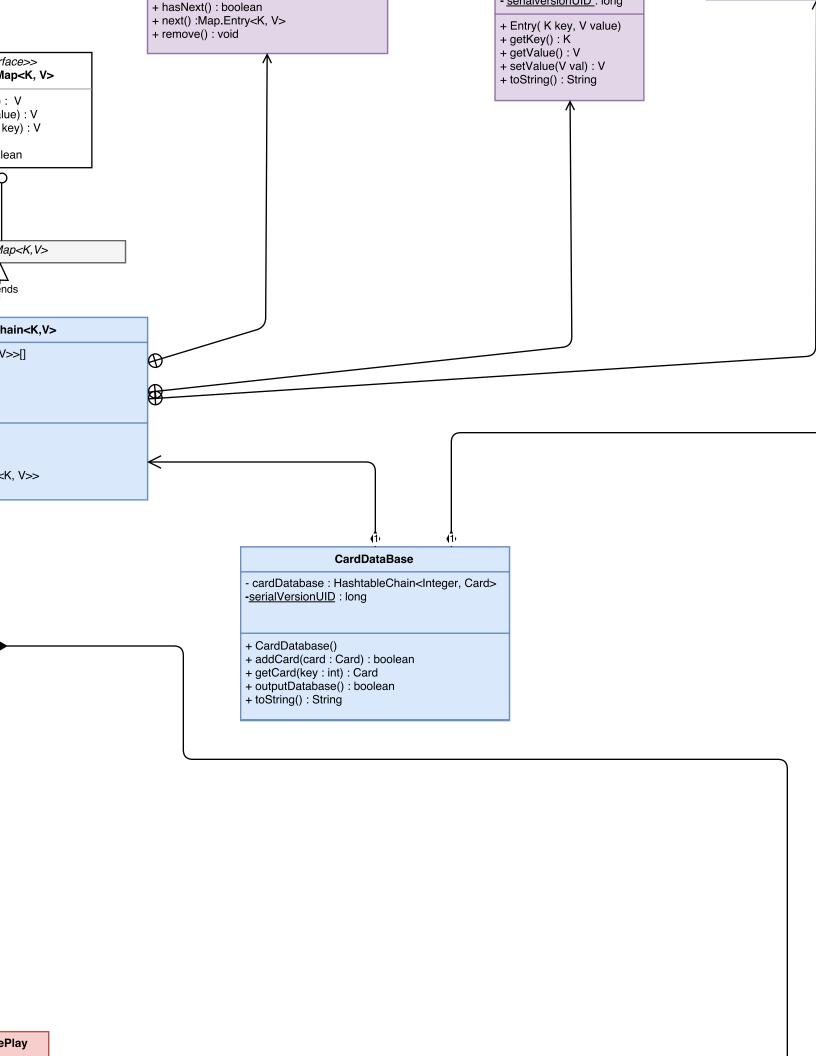
l

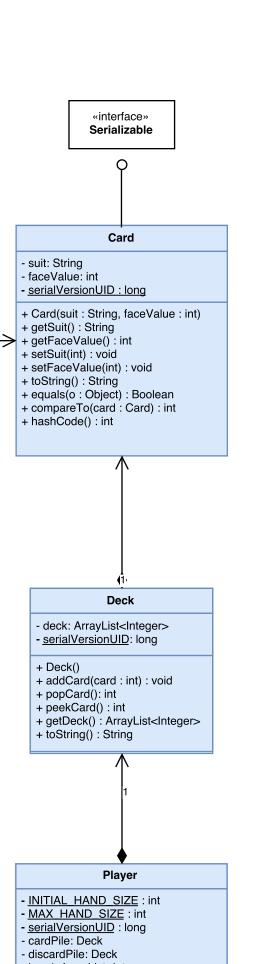
ySet

Map.Entry<K, V>>

+<u>login(uDataba</u> +<u>deckManger(u</u> +<u>deleteUser(uD</u> +<u>cardPickPlaye</u>

<<Inte KWHashI\ + get(Object key) + put(K key, V va + remove(Object + size(): int + isEmpty(): boo AbstractN Exte «interface» **HashTableC** Serializable - table : LinkedList<Entry<K, - numkeys : int - CAPACITY: int - LOAD\_THRESHOLD : int - serialversionUID : long + HashTableChain() + rehash(): void + toString(): String + entrySet() : Set<Map.Entry-**UserDataBase** - userDatabase : HashTableChain<String, Player> UserInterface + UserDatabase() se : UserDatabase, Scanner userInput) : void + getUser(key : String) : Player ser : Player, input : Scanner) : void + userExists(key : String) : boolean + addOrUpdateUser(key : String, player : Player) : boolean atabase: UserDatabase, userName: String, Scanner userInput): boolean r(handSize : int, Scanner userInput) : int + removeUser(key : String) : boolean + outputDataBase(): boolean + toString(): String QuickSort + sort(T[] inputArray, Comparator<T> comp): void Gam ort/T[] inputArray int lowValue int highValue





- bubbleSort3(T[] inputArray, int first, int last, Comparator<T> comp): void
  partition(T[] inputArray, int first, int last, Comparator<T> comp): int
  swap(Object[] inuputArray, int first, int last): void
  - SuitComparator
  - cards : CardDatabase
  - + compare(Integer left, Interger right)

## **FaceValueComparator**

- cards : CardDatabase
- + compare(Integer left, Integer right)

MakeDeck

+<u>createDeck(user : Player) : void</u>

+<u>shuffle(deck : ArrayList<Integer>) : void</u>

