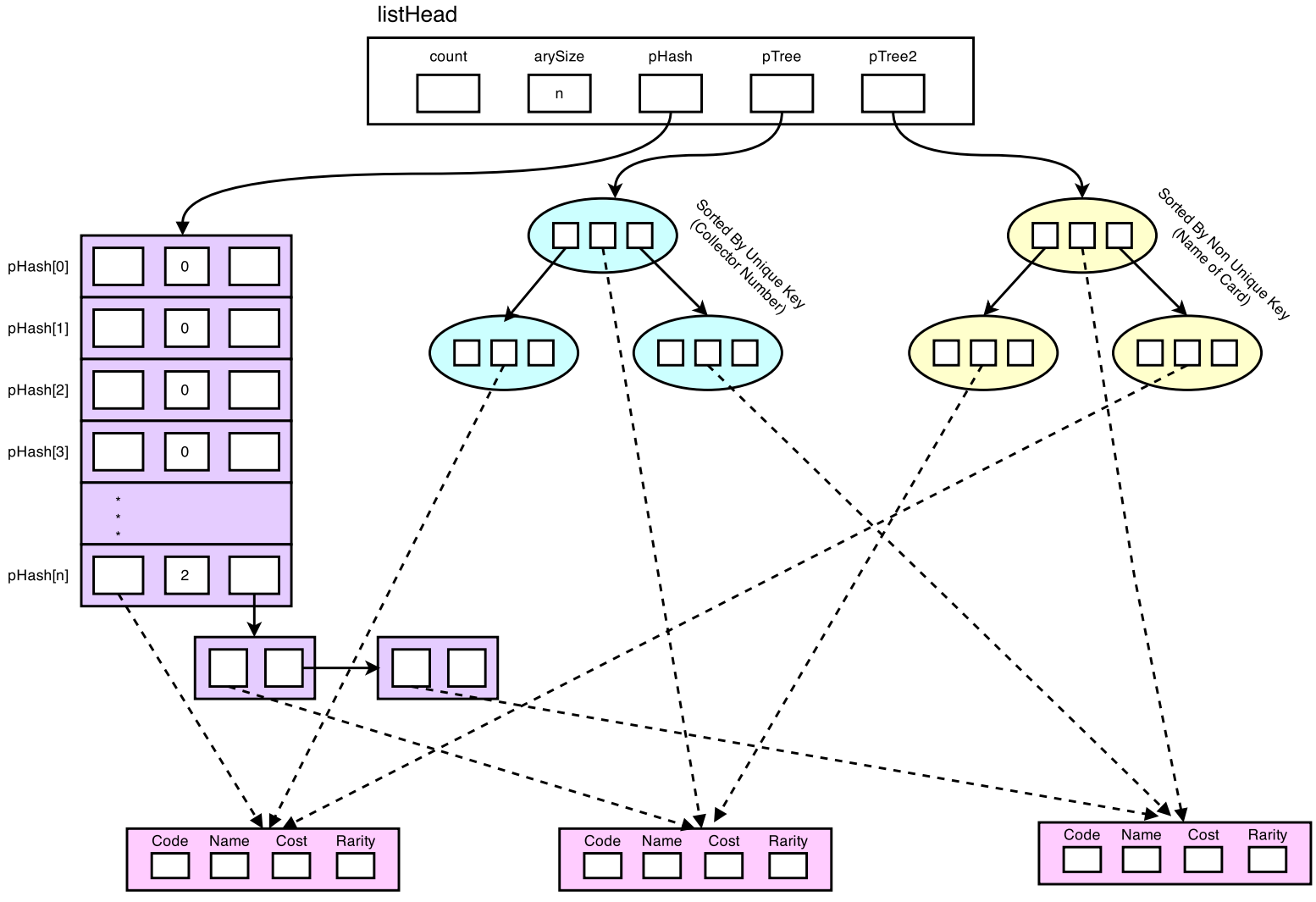
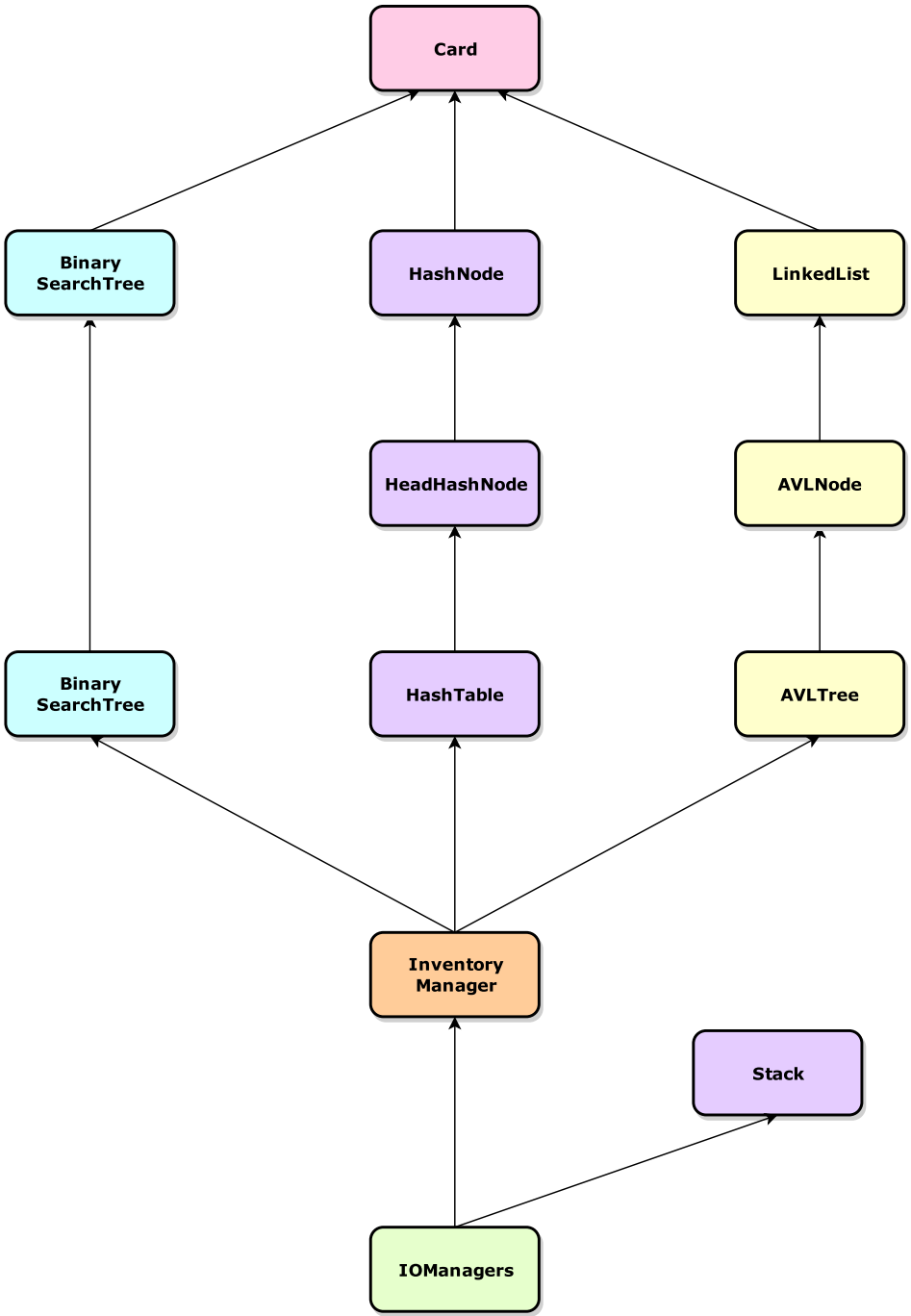
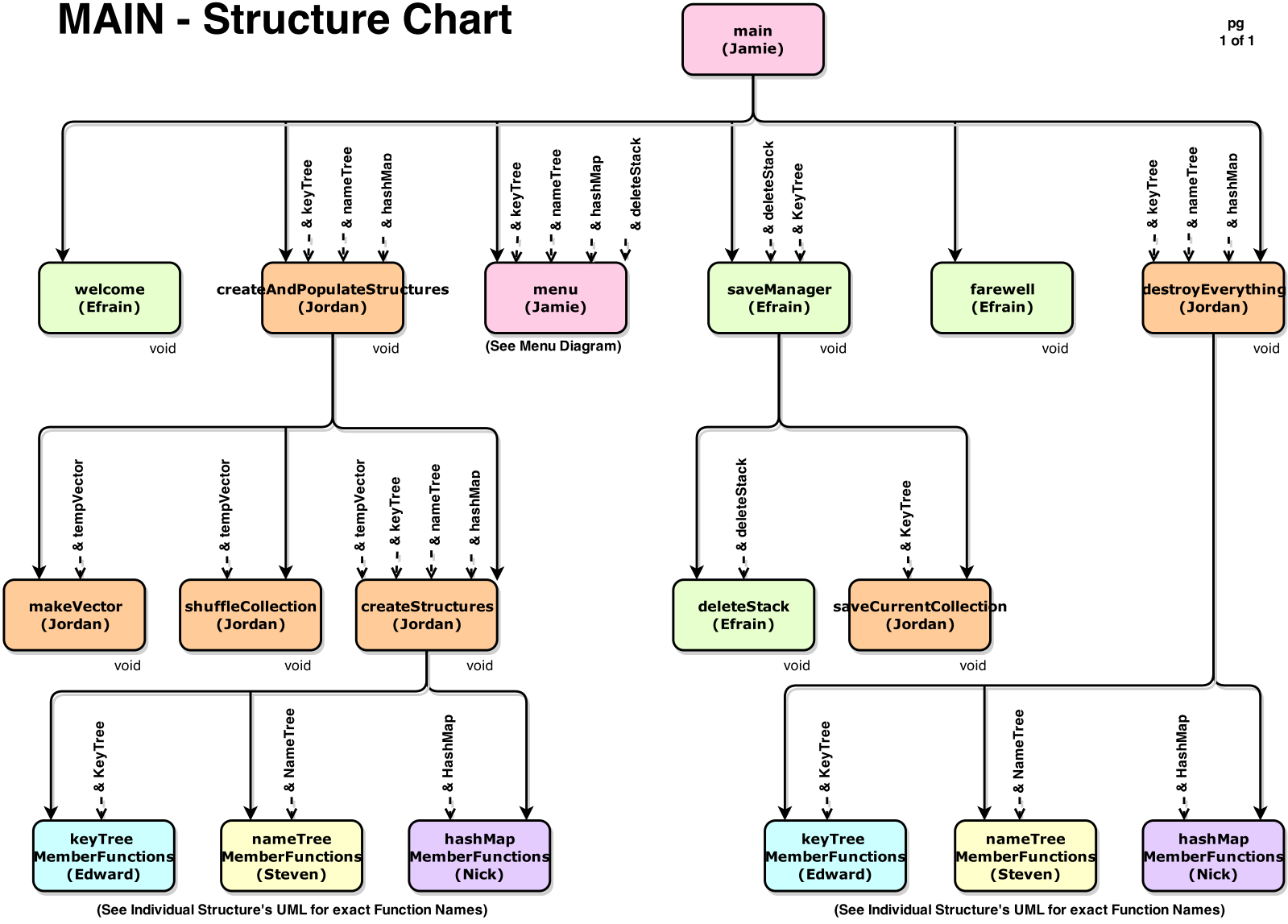

listHead
count
arySize
pHash
pTree
pTree2
n
pHash[0]
pHash[1]
pHash[2]
pHash[3]
pHash[n]
0
0
0
0
2
Sorted By Unique Key
(Collector Number)
Sorted By Non Unique Key
(Name of Card)
Code
Name
Cost
Rarity
Code
Name
Cost
Rarity
Code
Name
Cost
Rarity

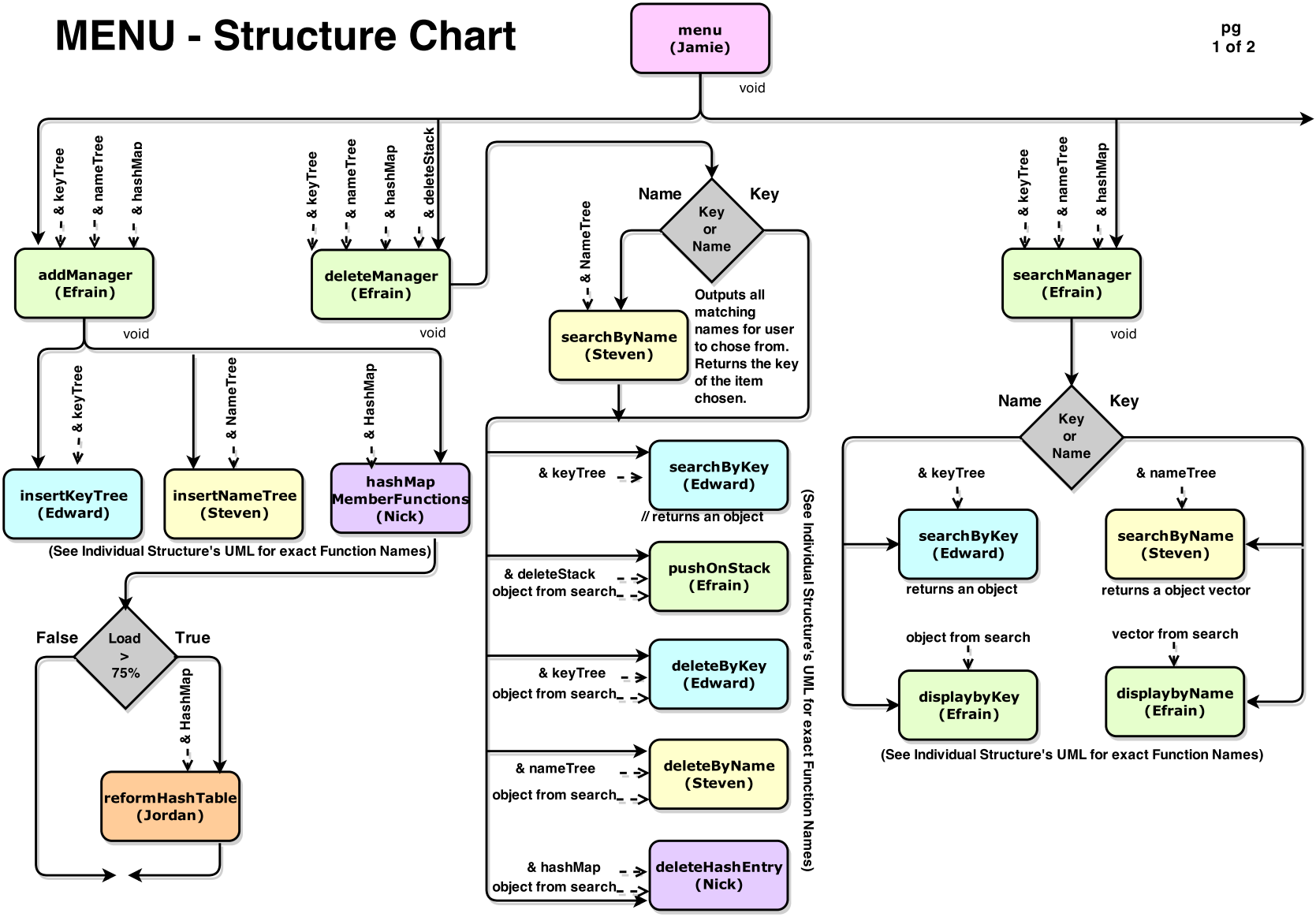
UML Structure Chart



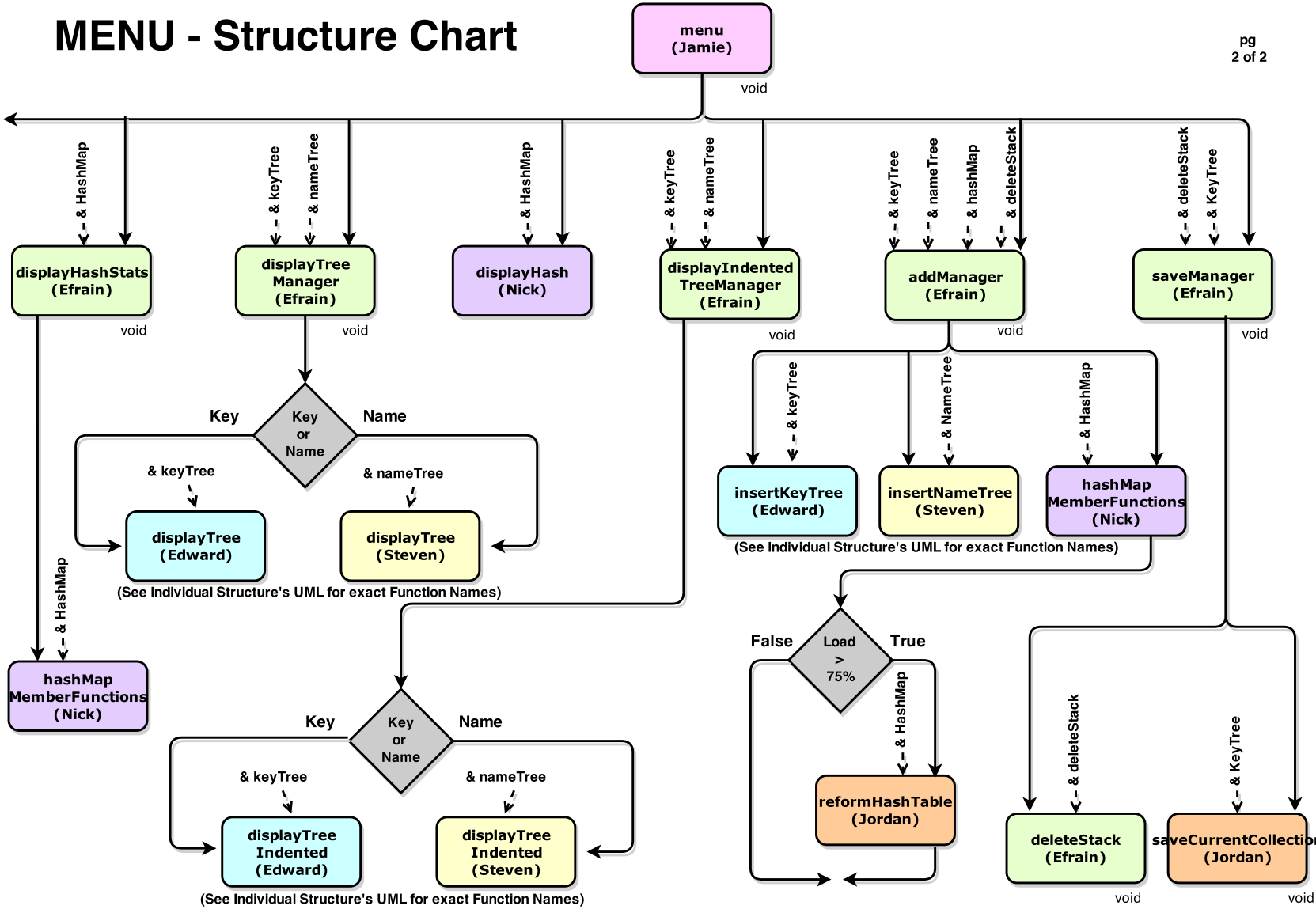
MAIN - Structure Chart



MENU - Structure Chart



MENU - Structure Chart



Hash - UML

Pt. 1 of 2

HashTable
-table: HeadHashNode<KeyType, ItemType>** -tableSize: int -filledSlots: int -collisions: int -loadFactor: double -listCount: int -longListCount: int -totalInLists: int -avgInLists: int
<<create>>-HashTable(size: int) <<create>>-HashTable(hash(const KeyType &, int): int, HashTable&, size: int) <<destroy>>-HashTable() -getHashValue(hash(const KeyType &, int): int, key: KeyType&) -init(size:int): void +addEntry(hash(const KeyType &, int): int, key: KeyType&, item: ItemType&): void +displayTable(display(ItemType&): void): void +printTable(display(ItemType&): void): void +search(hash(const KeyType &, int): int, key: KeyType&, item: ItemType&): bool +remove(hash(const KeyType &, int): int, key: KeyType&, item: ItemType&): bool +displayStatistics(): void +getTableSize(): int +getFilledSlots(): int +getCollisions(): int +getLoadFactor(): double +getListCount(): int +getTLongListCount(): int +getTotalInLists(): int +getAvgInLists(): int +getKeys(keys: vector<KeyType>&): void +getItems(items: vector<ItemType>&): int +destroyTable(): void

Hash - UML

Pt. 1 of 2

HashNode
-key: KeyType -item: ItemType -next: HashNode*
<<create>>-HashNode(key: KeyType&, item: ItemType&) *<<destroy>>-HashNode()* +setKey(k: KeyType&); void +setItem(i: ItemType&); void +setNext(n: HashNode*); void +getKey(kt: KeyType&); void +getItem(it: ItemType&); void +getNext(); HashNode*

HeadHashNode
-listCount: Int
<<create>>-HeadHashNode(key: KeyType&, item: ItemType&) <<destroy>>-HeadHashNode() +addToList(key: KeyType&, item: ItemType&): void +removeFromList(target: KeyType&, item: ItemType&) +getListCount(): int

AVLTree - UML

Pt. 1 of 2

AVLTree
rootPtr: AVLNode # count: int
<<create>>+AVLTree() <<create>>+AVLTree(tree: AVLTree&) <<operator>>+=(sourceTree: AVLTree&): AVLTree& *<<destroy>>+BinaryTree()* +isEmpty(): bool +size(): int +clear(): void +preOrder(): void +inOrder(): void +postOrder(): void +indentedList(): void +insert(newEntry: Card&): bool +remove(newEntry: Card&): bool +getEntry(target: Card&): LinkedList* +getCard(tardet: Card&): bool -destroyTree(nodePtr: AVLNode*): void -copyTree(nodePtr: AVLNode*): AVLNode* -_preorder(AVLNode*): void -_inorder(AVLNode*): void -_postorder(AVLNode*): void -_indentedList(AVLNode*, int, char): void -_insert(nodePtr: AVLNode*, newNode: AVLNode*): AVLNode* -_remove(targetNodePtr: AVLNode*, target: Card, success: bool&): AVLNode* -findMin(nodePtr: AVLNode*): AVLNode* -removeMin(nodePtr: AVLNode*): AVLNode* -findNode(treePtr: AVLNode*, target: Card&): AVLNode* -height(nodePtr: AVLNode*): unsigned char -bfactor(nodePtr: AVLNode*): int -fixHeight(nodePtr: AVLNode*): void -rotateRight(nodePtr: AVLNode*): AVLNode* -rotateLeft(nodePtr: AVLNode*): AVLNode* -balance(nodePtr: AVLNode*): AVLNode*

AVLTree - UML

Pt. 2 of 2

AVLNode
-item: LinkedList -height: unsigned char -leftPtr: AVLNode* -rightPtr: AVLNode*
<<create>>+AVLNode(anItem: Card*) <<create>>+AVLNode(anItem: LinkedList&, size: char, left: AVLNode*, right: AVLNode*) <<destroy>>-TreeNode() +setItem(anItem: LinkedList&): void +setHeight(size: unsigned char): void +setLeftPtr(left: AVLNode*): void +setRightPtr(right: AVLNode*): void +getItem(): LinkedList* +getHeight(): unsigned char +getLeftPtr(): AVLNode* +getRightPtr(): AVLNode* +isLeaf(); bool

BinarySearchTree - UML

BinarySearchTree

rootPtr: TreeNode*
count: int

<<create>>+BinarySearchTree()
<<create>>+BinarySearchTree(sourceTree: BinarySearchTree&)
<<operator>>+= (sourceTree: BinaryTree&): BinarySearchTree&
<<destroy>>-BinarySearchTree()
+isEmpty(): bool
+size(): int
+clear(): void
+insert(code: string): bool
+remove(code: string): bool
+findNode(target: string&): TreeNode*
+displayTree(): void
+displayIndentedTree(): void
+writeTreeToFile(outfile: ofstream&): void
-_insert(nodePtr: TreeNode*, newNode: TreeNode*): TreeNode*
-_remove(nodePtr: TreeNode*, target: string, success: bool&): TreeNode*
-_displayIndentedTree(nodePtr: TreeNode*, lineNum: int&): void
-destroyTree(nodePtr: TreeNode*): void
-copyTree(nodePtr: TreeNode*): TreeNode*
-_inorder(nodePtr: TreeNode*): void
-_writeTreeToFile(current_pointer: TreeNode*, outfile: ofstream&): void

TreeNode

-cardPtr: Card*
-leftPtr: TreeNode*
-rightPtr: TreeNode*

<<create>>-TreeNode()
<<destroy>>-TreeNode()
+setLeftPtr(left: TreeNode*): void
+setRightPtr(right: TreeNode*): void
+setCardPtr(card: Card*): void
+getLeftPtr(): TreeNode*
+getRightPtr(): TreeNode*
+getCardPtr(): Card*
+isLeaf(); bool

InventoryManager - UML

InventoryManager

-makeVector(vector<Card*>&, string): void
-populateStructures(BinarySearchTree*, AVLTree*, HashTable<string, Card*>*, vector<Card*> &): void
-reformHashTable(HashTable<string, Card*>* &);
-getHashSizePrime(int): int
-checkNotPrime(int): bool
-static bool getSaveFileName(string&): bool
-removeNonAlphaNumeric(string&): void
-txtCheck(string &): void
-makeSaveFile(BinarySearchTree*, string): void
-replaceOrNot(string&): bool
-ripEmUp(vector<Card*> &): void
+inventoryCreation(BinarySearchTree*, AVLTree*, HashTable<string, Card*>* &): void
+checkLoadFactor(HashTable<string, Card*>* &): void
+saveCurrentCollection(BinarySearchTree*): void
+destroyEverything(BinarySearchTree* &, AVLTree* &, HashTable<string, Card*>* &): void

IOMangers - UML

IOManagers

-upper(string &s): void
-validKey(string &key): bool
-option(): char
+addManager(BinarySearchTree*, AVLTree*, HashTable<string, Card*>* &): void
+searchManager(BinarySearchTree*, AVLTree*, HashTable<string, Card*>*): void
+deleteManager(BinarySearchTree*, AVLTree*, HashTable<string, Card*>*, stack<Card*>*): void
+undoDeleteManager(BinarySearchTree*, AVLTree*, HashTable<string, Card*>*, stack<Card*>*): void
+saveManager(BinarySearchTree* keyTree, stack<Card*>* deleteStack): void
+DeleteStack(stack<Card*>* deleteStack): void
+displayTreeManager(BinarySearchTree* keyTree, AVLTree* nameTree): void
+displayIndentedTreeManager(BinarySearchTree* keyTree, AVLTree* nameTree): void
+displayHashedTable(HashTable<string, Card*>* hashTable): void
+displayList(LinkedList &anItem): void

Misc Classes - UML

Card

-code: string
-name: string
-cost: string
-rarity: string

<<create>>+Card()
<<destroy>>+Card()
+setCode(code: string): void
+setName(name: string): void
+setCost(cost: string): void
+setRarity(rarity: string): void
+getCode(): string
+getName(): string
+getCost(): string
+getRarity(): string
+operator<<(os: ostream&, card: Card&): ostream&
+operator<<(os: ostream&, card: Card*): ostream&
+operator>(other: Card&): bool
+operator<(other: Card&): bool
+operator>=(other: Card&): bool
+operator<=(other: Card&): bool
+operator==(other: Card&): bool
+operator!=(other: Card&): bool
+oat_hash(key: string&, num:int): static unsigned int
+display(card: Card*): void

LinkedList

-ListNode: struct
-data: ListNode.Card*
-next: ListNode.ListNode*
-head: ListNode*
-curr: ListNode*
-count: int

<<create>>+LinkedList()
<<destroy>>+LinkedList()
+GetCount():int
+IsEmpy(): bool
+ResetCurr(): void
+Search(target: Card&): bool
+GetNext(nextNode: Card*): bool
+GetFirst(nextNode: Card*): bool
+Insert(newData: Card&): bool
+Delete(delData: Card&): bool

Stack

-StackNode: struct<class T>
  value: T
  next: StackNode*
-top: StackNode*
-count: int

<<create>>+Stack()
<<destroy>>+Stack()
+push(T): bool
+pop(T &): bool
+pop(): bool
+isEmpty(): bool
+getTop(T &): bool
+getCount(): int