Robot

- serialnumber: string

- model: string

- alias: string

- comment: string

- productiondate: string

//constructor

+ Robot(string, string, string, string, string)

+ Robot(const Robot &obj)

//destructor

+ getSerialNumber(): String

+ getModel(): String

+ getAlias(): String

+ getComment(): String

+ getProductionDate(): String

+ setAlias(string): void

+ setComment(string): void

+ setProductionDate(string): void

+ operator==(const Robot &r) const: bool

+ operator<(const Robot &r) const: bool

+ operator>(const Robot &r) const: bool

// friend

+ operator<<(ostream &, const Robot &): ostream &

<templete ItemType>
HashTable

- table: ItemType**

- _sentinel: void*

- _size: int32_t

- _count: int32_t

- (*_hashFunction)(const ItemType &, const int32_t &): int32_t

---

//constructor

+ HashTable(int32_t, int32_t hashFunction(const T &, const int32_t &))

//destructor

+ size(): int32_t

+ count(): int32_t

+ at(int32_t): ItemType*

+ print(): int

+ insert(ItemType &): int32_t

+ hash(const ItemType &): int32_t

+ find(const ItemType &): int32_t

+ remove(const int32_t &): ItemType*

<templete ItemType>
Queue

- front: QueueNode*

- rear: QueueNode*

- count: int

//constructor

//destructor

+ enqueue(ItemType): bool

+ dequeue(ItemType &): bool

+ isEmpty(): bool

+ getCount(): int

+ queueFront(ItemType &): bool

+ queueRear(ItemType &): bool

Extends

<templete ItemType>
Binary Tree

- rootPtr: BinaryNode*<ItemType>

- count: int

//constructor

//destructor

+ isEmpty(): bool

+ size(): int

+ clear(): void

+ preOrder(void visit(ItemType&)): void

+ postOrder(void visit(ItemType&)): void

+ inOrder(void visit(ItemType&)): void

+ breadthOrder(void visit(ItemType&)): void

+ virtual insert(ItemType&): bool

+ virtual remove(ItemType&): bool

+ virtual getEntry(ItemType&): bool

- destroyTree(BinaryNode<ItemType>*): void

- _preOrder(void visit(ItemType&, BinaryNode<ItemType>*)): void

- _inOrder(void visit(ItemType&, BinaryNode<ItemType>*)): void

- _postOrder(void visit(ItemType&, BinaryNode<ItemType>*)): void

- _breadthOrder(void visit(ItemType&, BinaryNode<ItemType>*)): void

<templete ItemType>
Binary Search Tree

- root: BinaryNode*

- count: int

- (*comp)(ItemType &, ItemType &): int

//constructor

BinarySearchTree(int compare(ItemType &, ItemType &))

//destructor

+ insert(ItemType&): bool

+ remove(ItemType&): bool

+ getEntry(const ItemType, &ItemType&): bool

+ getMinEntry(ItemType&): bool

+ getMaxEntry(ItemType&): bool

- _insert(BinaryNode*, BinaryNode*): BinaryNode*

- _remove(BinaryNode*, ItemType, bool): BinaryNode*

- deleteNode(BinaryNode*): BinaryNode*

- findNode(BinaryNode*, ItemType&): BinaryNode*

<templete ItemType>
Binary Node

- item: ItemType

- leftPtr: BinaryNode<ItemType>*

- rightPtr: BinaryNode<ItemType>*

//constructor

//destructor

+ setItem(ItemType&): void

+ setLeftPtr(BinaryNode<ItemType>*): void

+ setRightPtr(BinaryNode<ItemType>*): void

+ getItem(): ItemType

+ getLeftPtr(): BinaryNode<ItemType>*

+ getRightPtr(): BinaryNode<ItemType>*

+ isLeaf():bool