BagInterface

```java
package Interfaces;

import main.Card;

/**
 * Generic Type of Bag Interface, for this application.
 * Will be Used in the CardSlotsBag class.
 *
 * @param <T> the Type of the Bag
 */
public interface BagInterface<T> {

    /**
     * get the current Size of the Bag.
     *
     * @return size of the bag
     */
    public int getCurrentSize();

    /**
     * Check if the bag is empty or not
     *
     * @return returns true if empty false if not
     */
    public boolean isEmpty();

    /**
     * Add a new entity to the bag
     *
     * @param anEntry the entity to add.
     * @return returns true if added or false if not
     */
    public boolean addNewEntry(T anEntry);

    /**
     * Removes the last entity in the bag.
     *
     * @return
     */
    public T remove();

    /**
     * Removes an entity if it can find the input entity
     *
     * @param anEntry the entity to remove if can be found.
     * @return the Card found and removed.
     */
    public Card remove(T anEntry);

    /**
     * Clear the Bag.
     */
    public void clear();

    /**
     * Check if the bag contains the given entity.
     *
     * @param anEntry the entity to find
```

```
     * @return returns true if found, false if not.
     */
    public boolean contains(T anEntry);

    /**
     * Convert the Bag to an array.
     *
     * @return converts the bag into any array.
     */
    public T[] toArrayCopy();

}
```

StackInterface

```
package Interfaces;

/**
 * A Generic Type of Stack to be used in this application.
 * Will be used in the Deck Class.
 *
 * @param <T> the Type of the Stack.
 */
public interface StackInterface<T> {

    /**
     * Push a new entity on to the top of the stack.
     *
     * @param newEntry entity to add
     */
    public void push(T newEntry);

    /**
     * Remove the entity at the stop of the stack and return it.
     *
     * @return Object at the top of the stack.
     */
    public T pop();

    /**
     * Returns the Object at the top of the stack but does not remove it.
     *
     * @return object at the top of the stack.
     */
    public T peek();

    /**
     * Checks if the stack is empty or not.
     *
     * @return returns true if the stack is empty
     */
    public boolean isEmpty();

    /**
     * Clears the stack
     */
    public void clear();
```

```
}
```

QueueInterface

```
package Interfaces;

/**
 * A Generic Type of a Queue ADT, for use in this project.
 * With be used in the RoundQueue.
 *
 * @param <T> the type of the stack.
 */
public interface QueueInterface<T> {

    /**
     * Add a entity to the back of the que or front and back if its the
first object in the Queue.
     *
     * @param newEntry the entity to add to the back of the queue
     */
    public void enqueue(T newEntry);

    /**
     * Removes the entity at the front of the queue
     *
     * @return returns the Object that was dequeued
     */
    public T dequeue();

    /**
     * Returns but does not remove the Object at the front of the queue.
     *
     * @return returns Object at the front of the queue.
     */
    public T getFront();

    /**
     * Check if the Queue is empty or not
     *
     * @return returns true if empty, false if not
     */
    public boolean isEmpty();

    /**
     * Clears the Queue.
     */
    public void clear();

}
```