**ADT as a Blueprint (Interface in Java)**

An ADT is usually defined as an **interface** in Java, which means:

* It **only declares** method signatures.
* It **does not contain** any implementation details.

For example, here’s an **ADT Bag** interface:

public interface BagInterface<T> {

int getCurrentSize();

boolean isEmpty();

boolean addNewEntry(T newEntry);

T remove();

boolean remove(T anEntry);

void clear();

boolean contains(T anEntry);

T[] toArray();

}

**Different Implementations of the Same ADT**

Since ADTs **do not dictate how** they should be implemented, a developer can implement the same ADT using different data structures.

**🔹 Implementation 1: Bag Using a Fixed-Size Array**

import java.util.Arrays;

public class ArrayBag<T> implements BagInterface<T> {

private final T[] bag;

private int numberOfEntries;

private static final int DEFAULT\_CAPACITY = 10;

public ArrayBag() {

this(DEFAULT\_CAPACITY);

}

@SuppressWarnings("unchecked")

public ArrayBag(int capacity) {

bag = (T[]) new Object[capacity];

numberOfEntries = 0;

}

@Override

public boolean addNewEntry(T newEntry) {

if (numberOfEntries < bag.length) {

bag[numberOfEntries++] = newEntry;

return true;

}

return false; // Bag is full

}

@Override

public T remove() {

if (!isEmpty()) {

T result = bag[numberOfEntries - 1];

bag[numberOfEntries - 1] = null;

numberOfEntries--;

return result;

}

return null;

}

@Override

public boolean remove(T anEntry) {

for (int i = 0; i < numberOfEntries; i++) {

if (bag[i].equals(anEntry)) {

bag[i] = bag[numberOfEntries - 1];

bag[numberOfEntries - 1] = null;

numberOfEntries--;

return true;

}

}

return false;

}

@Override

public int getCurrentSize() {

return numberOfEntries;

}

@Override

public boolean isEmpty() {

return numberOfEntries == 0;

}

@Override

public boolean contains(T anEntry) {

for (int i = 0; i < numberOfEntries; i++) {

if (bag[i].equals(anEntry)) {

return true;

}

}

return false;

}

@Override

public void clear() {

while (!isEmpty()) {

remove();

}

}

@Override

public T[] toArray() {

return Arrays.copyOf(bag, numberOfEntries);

}

}

**Testing Both Implementations**

We can now create a **BagDemo** program to test both implementations:

public class BagDemo {

public static void main(String[] args) {

BagInterface<String> arrayBag = new ArrayBag<>(5);

BagInterface<String> linkedBag = new LinkedBag<>();

arrayBag.addNewEntry("Apple");

arrayBag.addNewEntry("Banana");

linkedBag.addNewEntry("Carrot");

linkedBag.addNewEntry("Orange");

System.out.println("Array Bag Contains 'Apple'? " + arrayBag.contains("Apple"));

System.out.println("Linked Bag Contains 'Carrot'? " + linkedBag.contains("Carrot"));

arrayBag.remove("Apple");

linkedBag.remove();

System.out.println("Array Bag Size: " + arrayBag.getCurrentSize());

System.out.println("Linked Bag Size: " + linkedBag.getCurrentSize());

}

}