List61B, **SLList**, and **AList**: Reference Sheet, Page 1

```
public interface List61B<T> {
    public void addLast(T x);
    public T getLast();
    public T get(int i);
    public int size();
    public T removeLast();
    public void insert(T x, int position);
    public void addFirst(T x);
    public T getFirst();
}

public class SLList<T> implements List61B<T> {
    private class Node {
        T item;     /* Equivalent of first */
        Node next; /* Equivalent of rest */
        Node(T i, Node n) {
            item = i;
            next = n;
        }
    }
    private Node sentinel;
    private int size;
    public SLList() {
        sentinel = new Node(null, null); size = 0;
    }
    ...
}


public class AList<T> implements List61B<T> {
    private T[] items;
    private int size;
    public AList() {
        items = (T[]) new Object[8]; size = 0;
    }
    private void resize(int s) { ... }
    ...
}
```

IntList:

```
public class IntList {
    public int first;
    public IntList rest;

    public IntList(int f, IntList r) {
        first = f;
        rest = r;
    }
    /** Returns an IntList with the given numbers.*/
    public static IntList of(int[] input) { ... }
}
```

JUnit methods:

```
assertEquals(Object x, Object y)
assertEquals(int x, int y)
assertEquals(double x, double y)
assertTrue(boolean b)
assertFalse(boolean b)
assertNotNull(Object x)
assertArrayEquals(Object[] x, Object[] y)
assertArrayEquals(int[] x, int[] y)
assertArrayEquals(double[] x, double[] y)
```

Iterator, **Iterable**, **Comparator**, **Comparable**:

```
public interface Iterator<T> {
    boolean hasNext();
    T next();
}

public interface Iterable<T> {
    Iterator<T> iterator();
}

public interface Comparator<T> {
     int compare(T o1, T o2);
}

public interface Comparable<T> {
     int compareTo(T obj);
}
```