CS62: Fall 2025 | Lecture #4 (Interfaces, ArrayLists) worksheet | Prof. Li

1.a) On a research expedition studying air traffic, we discovered a new species: the Flying Interfacing **CatBus**, which acts like a vehicle and has the ability to make noise. Given the Vehicle and Noisemaker interfaces, fill out the CatBus class so that CatBuses can rev their engines and make noise at other CatBuses with a CatBus-specific sound.

```
interface Vehicle {
    public void revEngine();
}
interface Noisemaker {
    public void makeNoise();
}
public class CatBus ____________________ _________________, _________________ {
  @Override
  __________ __________ _____________ { /* CatBus revs engine, code not shown */ }
  @Override
  __________ __________ _____________ { /* CatBus makes noise, code not shown */ }
  /** Allows CatBus to make noise at other CatBuses. */
  public void conversation(CatBus target) {
    makeNoise();
    target.makeNoise();
  }
}
```

1.b) We've encountered a **Goose** in the skies, which also implements **Noisemaker**. Modify the conversation method signature so that **CatBuses** can makeNoise at **both** CatBus and Goose objects while still only having one argument, **target**.

2) Fill in the implementation for add(int I, E element) in the right column. The left column is other methods for reference.

```java
public class ArrayList<E> implements List<E> {
    private E[] data; // underlying array of Es
    private int size; // number of Es

    public ArrayList() {
        data = (E[]) new Object[2];
        size = 0;
    }

    public ArrayList(int capacity) {
        data = (E[]) new Object[capacity];
        size = 0;
    }

    public boolean isEmpty() {
        return size == 0;
    }

    public int size() {
        return size;
    }

    private void resize(int capacity) {
        E[] temp = (E[]) new Object[capacity];
        for (int i = 0; i < size; i++){
            temp[i] = data[i];
        }
        data = temp;
    }

    public void add(E element) {
        if (size == data.length){
            resize(2 * data.length);
        }

        data[size] = element;
        size++;
    }
    public E set(int index, E element) {
        if (index >= size || index < 0){
            throw new
IndexOutOfBoundsException("Index " + index + "
out of bounds");
        }
        E old = data[index];
        data[index] = element;
        return old;
    }

    public E get(int index) {
        if (index >= size || index < 0){
            throw new
IndexOutOfBoundsException("Index " + index + "
out of bounds");
        }

        return data[index];
    }
```

```java
    /**
     * Inserts the element at the specified
index. Shifts existing elements to the
     * right and doubles its capacity if
necessary.
     *
     * @param index the index to insert the
element
     * @param element the element to be inserted
     * @pre: 0 <= index <= size
     */
    public void add(int index, E element) {
        //TODO: check whether index is in range




        //TODO: if full double in size




        //TODO: shift elements to the right




        //TODO: increase number of elements




        //TODO: put the element at the right
index in data



    }
```