What if our Frog (from lab) could say 10 different things?

public class Frog{
    private boolean formerPrince;
    private String phrase1;
    private String phrase2;
    private String phrase3;
    private String phrase4;
    private String phrase5;
    private String phrase6;
    private String phrase7;
    private String phrase8;
    private String phrase9;
    private String phrase10;
    ...
}

What if a Person could adopt lots of Pets?

public class Person{
    private String name;
    private Pet pet1;
    private Pet pet2;
    private Pet pet3;
    private Pet pet4;
    private Pet pet5;
    private Pet pet6;
    private Pet pet7;
    private Pet pet8;
    private Pet pet9;
    private Pet pet10;
    private Pet pet11;
    private Pet pet12;
    private Pet pet13;
    private Pet pet14;
    private Pet pet15;
    ...
}

Array of Primitives

int[] data;
data = new int[3];
data[0] = 5;
data[1] = 10;
Array of Objects

Counter[] counters;

counters = new Counter[3];
- it has 3 references to Counters in it
- all of these references are initially null

counters[0] = new Counter();
counters[0].addOne();
counters[1] = new Counter();
counters[0].getCount() -> 1

counters[2].getCount() -> NullPointerException

E.g. Person Database

public class Person{
    private String name;
    private int age;

    Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    public int getAge() { return age; }
    public String getName() { return name; }
}

Person Array

Person[] people = {new Person("jo"), new Person("flo")};

E.g. Person Database (contd..)

public class PersonDB{
    private Person[] people;

    public PersonDB(Person[] p){
        people = p;
    }

    /** Calculates and returns the average age. */
    public int getAverageAge(){
        int sum = 0;
        for(int i = 0; i < people.length; i++){
            sum = sum + people[i].getAge();
        }
        return (sum/people.length);
    }
}

Person[] people = new Person[] {new Person("jo"),
                                new Person("flo")};

Person[] people;

people = new Person[3];

ESE112

people[0] = new Person("jo");
people[1] = new Person("flo");

ESE112

people[0] = new Person("jo");
people[1] = new Person("flo");

ESE112

ESE112
Error Checking – To avoid runtime errors
/** Calculates and returns the average age. */
public int getAverageAge(){
    int sum = 0;
    int numOfEntries = 0;
    if(people == null || people.length == 0) {
        return sum;
    }
    for(int i = 0; i < people.length ; i++){
        if(people[i] != null){
            sum = sum + people[i].getAge();
            numOfEntries++;
        }
    }
    return (sum/numOfEntries);
}

Note on Comparing Strings
Since String is reference type
- Comparing variables of type String compares the heap addresses
  - String s1 = “hello”;
  - String s2 = “hello”;
  - s1 == s2 results in false
- If you want to compare content use the equals(..)/equalsIgnoreCase(..) method in the String class

Complete in PersonDB
/** Returns true if name is in database, otherwise false
   Note: Recollect what we said about comparing strings */
public boolean isInDatabase(String searchName){
}