Grouping Objects into Flexible-size Collections

1 We know Arrays

As we have seen in C, we have the possibility to list elements into an array. This is one of our **primitive data structures**.

You can do the same thing in Java:

Create a new int array with a length of 7:

```
numbers = new int[7];
```

And we can assign values to each int-element in the array:

```
numbers[0] = 6;
numbers[1] = 19;
```

etc.

2 Objects in an Array

We can also create arrays and add objects:

Just say, you've got 3 dogs at home:

```
Dog[] myDogs = new Dog[3];   //size of elements

            myDogs[0] = new Dog("lupo", 12);
            myDogs[1] = new Dog("fido", 4);
            myDogs[2] = new Dog("sepp", 7);
```

Now we can access each dog through the array:

```
myDogs[1].bark();
```

See Dog example

Note: once you have created an array of a certain type, you can't put anything in it. You couldn't put a Cat into a Dog.

3 Collections

The Java class libraries have Collection classes which extend the basic idea of a data structure. They can be pretty powerful when you have to deal with objects.

The basic thing is: a collection is an object which allows you to store other objects.

A typical example is the **ArrayList**.

```
ArrayList<Dog> myDogs;
myDogs = new ArrayList<Dog>();
```

➔ **Note: the array list is like a flexible container. So we don't have to define a fixed size.**

This sounds a bit like the "linked list" in C, remember ? There we tried to be flexible with the size as well.

With the *myDogs* object you can do all sorts of operations. It has **a set of methods** – unlike a primitive array which is not an object.

Resarch & Questions:

Try out different collection classes with the Dog example!
What can you do with an ArrayList? Check out the methods in the API….