Java arrays, like C arrays, are more limited than their counterparts in Python or Javascript. Java arrays cannot be resized once created, and you must declare the type of the variables that the array will store in advance. The `ArrayList` class is more flexible, and a frequent alternative to arrays, but first let's see how to use arrays.

An example of using arrays:

```
package com.github.akarazhev.jacademy.jprog.basics;

public final class ArrayExample {

    public static void main(final String[] args) {
        // an array of ints
        final int[] myNumbers = {10, 15, 20, 25, 30};

        System.out.println(myNumbers[2]);

        // arrays have an instance variable that stores
        //  the length of the array:
        System.out.println(myNumbers.length);

        // Arrays can be of any type a variable can be:
        final String[] myStrings = {"Narnia", "Oz", "Neverland"};
        System.out.println("The Wizard of " + myStrings[1]);

        // arrays can be modified
        myStrings[2] = "Wonderland";
    }
}
```

Notice that although you can determine the length of a string using the method `someString.length()`, you determine the length of an array using the instance variable `someArray.length`. I consider this inconsistency to be a design flaw in Java; you'll just have to remember it.

Declaring arrays

Like any other variable, an array must be declared, and its type specified. The type for an array of `int` values is `int[]`. You can think of the empty brackets as shorthand for the word "array":

```
int[] myNumbers;
```

This is only a declaration of the variable that will hold the array. The array does not yet exist and so does not have a length.

Initializing arrays: shortcut notation

We frequently want to give an array some initial values. We have seen the shortcut method already:

```
int[] myNumbers = {10, 15, 20, 25, 30};
```

The shortcut method automatically creates an array of the neccessary length, and initializes the array with the given values. Be careful: the length of `myNumbers` in the example is now 5, and cannot be increased. Shortcut notation can only be used at the time the array is declared.

Initializing arrays with `new`

An array, like a String, is a special type of built-in object. The code

```
int[] myNumbers;
```

declares the variable `myNumbers`, but does not yet create the array object. The special keyword `new` in Java is used to create objects, including arrays.

```
package com.github.akarazhev.jacademy.jprog.basics;

public final class NewArray {

    public static void main(final String[] args) {
        final int[] myNumbers = new int[5];

        myNumbers[0] = 10;
        myNumbers[1] = 10;
        myNumbers[2] = 10;
        myNumbers[3] = 10;
        myNumbers[4] = 10;

        System.out.println("The array myNumbers has length " + myNumbers.length);
    }
}
```

The array in `main (String[] args)`

You can now see that `main` takes a single parameter, `args`, of type String array. How is it used? If you execute a program from the command line (terminal on Unix or Mac), you can pass parameters to the program. For example, the command

```
wc README.txt
```

could be typed into a unix terminal to count the words in README.txt. The first word, `wc`, is the name of the program to execute. The second word, `README.txt`, is passed as a string as the first item in the array of strings, `args`.

A similar mechanism is used in C, but since C arrays do not know their lengths, the syntax of the declaration of main is a bit different. In Python, `sys.argv` can be imported to fetch command-line arguments.

The `java.util.Arrays` class

Although arrays in Java are objects, they are built-in, and and as mentioned, are not very flexible. The `Arrays` class in the `java.util` package provides some methods that let you do things like sort, binary search, or print a string representation of an array. Unlike some other classes, like `Math` or `String` that are available by default, you must request that Java make the `Arrays` class available to you, using an *import statement* at the top of any file in which you would like to use it:

```
package com.github.akarazhev.jacademy.jprog.basics;

import java.util.Arrays;

public final class ArraysExample {

    public static void main(final String[] args) {
        final int[] myNumbers = {42, 1, 17, 27, 16};
        Arrays.sort(myNumbers);
        System.out.println(Arrays.toString(myNumbers));
    }
}
```

Notice that `Arrays.sort` works in place. Also notice that there are several overloaded `toString` methods that work on arrays containing ints, doubles, and even objects. There is no `reverse` method; you'll have to write your own or use an `ArrayList` object to store your list of data.