Arrays.

A data structure for storing homogenous data sets (i.e. sets of same type)

Problem: Exam grade average:

Ask the user to enter how many grades there are. Then ask them to enter the grades. Compute the average grade, and how many grades are above the average.
Syntax for declaring an array:

\[<\text{typename}>[] \text{Array\_Name} = \text{new} <\text{typename}>[\text{Length}]\;\]

Examples:

- // 80-element array with base type \text{char}:
  \begin{align*}
  \text{char[]} \text{symbol} &= \text{new} \text{char}[80]; \\
  \end{align*}

Diagrammatically:

\begin{figure}
\centering
\includegraphics[width=\textwidth]{symbol.png}
\caption{Diagram of a 80-element char array.}
\end{figure}

- // 100-element array of doubles:
  \begin{align*}
  \text{size} &= 100 \\
  \text{double[]} \text{reading} &= \text{new} \text{double}[\text{size}]; \\
  \end{align*}

Notice the use of [ ]'s above.

Array indexing and length:

Refer to a specific element in the array (also called an indexed variable) by using a subscript (a.k.a. index) in [ ].

Array subscripts use zero-numbering:

- the first element has subscript 0,
- ... the last element has subscript \text{length-1}

Example:

\begin{figure}
\centering
\includegraphics[width=\textwidth]{example.png}
\caption{Example of array indexing in code.}
\end{figure}

Length of an array is specified by the number in brackets when it is created with operator \text{new}:

- it determines the maximum number of elements the array can hold.
--storage is allocated whether or not the elements are assigned values

To get the length of array add .length to the name, e.g. if

    int [] grade = new int[25];
    
    grade.length refers to the number of elements in grade, i.e. 25

    (Notice the absence of parenthesis after the word length)

The length attribute is established in the declaration and cannot be changed unless the array is redeclared.

Array Initialization:

1. Array elements can be initialized in the declaration statement by putting a comma-separated list in braces as shown on the example below.
   The length of an array is automatically determined when the values are explicitly initialized in the declaration
   Example:
       double[] reading = {5.1, 3.02, 9.65};
       System.out.println(readings.length);

       Q: what will be displayed?

2. More often, initialization (and other kinds of loop processing) is done in a loop.
   Loop counter is often used as an array subscript(index)

   int i;//loop counter/array index
   int[] a = new int[10];
   
   for(i = 0; i < a.length; i++)
       a[i] = 0;
Array and array elements as method arguments.

Entire arrays and individual array elements can be passed to methods.

a. Array elements as actual parameters – are treated as any other variable, i.e. their value is passed in.

Example: Q: What will be printed as the result of execution of main?

```java
public static void multiplyBy3 (double x) {
    x = x*3;
    System.out.println ("x is "+x);
}

public static void main (String [] args) {
    double [] a = {10, 20, 30};
    System.out.print (a[0]);
    System.out.print (a[1]);
    System.out.print (a[2]);
    multiplyBy3(a[1]);
    System.out.print (a[1]);
}

Answer: 102030
x is 60
20
```

b. Array as a parameter.

Notice the syntax in the parameter list of the method:

```java
public static void showArray(char[] array){
    int i;
    for(i = 0; i < a.length; i++)
        System.out.println(array[i]);
}
```

in the method invocation array is passed via the array variable name, no brackets!:

```java
showArray (a)  // in main module
```
Subtle point: when an entire array is passed into the method and the method changes the array elements, the changes will also take place outside the method – see example

Example

```java
public static void giveRaise (double[] salary, int amt) {
    int i;
    for(i = 0; i < salary.length; i++) {
        salary [i] = salary[i] + amt;
    }
}

public static void main (String [] args) {
    double [] emp_salary = {50000.90, 20000.40, 30000.50};
    System.out.print (emp_salary[0]);
    System.out.print (emp_salary[1]);
    System.out.print (emp_salary[2]);
    giveRaise(emp_salary, 5000);
    System.out.print (emp_salary[0]);
    System.out.print (emp_salary[1]);
    System.out.print (emp_salary[2]);
}
```

Will print

```
50000.90 20000.40 30000.50
55000.90 25000.40 35000.50
```
Same subtlety in assignments (=)

Example

```java
int[] a = new int[3];
int[] b = new int[3];

for(int i; i < a.length; i++)
    a[i] = i;

b = a;

System.out.println(a[2] + " " + b[2]);
a[2] = 10;
System.out.println(a[2] + " " + b[2]);
```

The output for this code will be:

```
2 2
10 10
```

Same subtlety in comparisons (==):

```java
int i;
int[] a = new int[3];
int[] b = new int[3];

for(i=0; i < a.length; i++)
    a[i] = i;

for(i=0; i < b.length; i++)
    b[i] = i;

if(b == a)
    System.out.println("a equals b");
else
    System.out.println("a does not equal b");
```

The output for this code will be:

```
a does not equal b
```
Arrays as method return values:

Example:

```java
public class returnArrayDemo {
    public static void main(String arg[]) {
        char[] c;
        c = vowels();
        for(int i = 0; i < c.length; i++)
            System.out.println(c[i]);
    }

    public static char[] vowels() {
        char[] newArray = new char[5];
        newArray[0] = 'a';
        newArray[1] = 'e';
        newArray[2] = 'i';
        newArray[3] = 'o';
        newArray[4] = 'u';
        return newArray;
    }
}
```