7.2 Arrays, Classes, and Methods

- Arrays
  - » can be used as instance variables in classes
  - » Both an indexed variable of an array and an entire array can be an argument to a method
  - » methods can be return an array value.
  - » Arrays can be used with classes and methods just as other objects can.

Listing 7.3 Sales Associate

```
import java.util.Scanner;

/**
 Class for sales associate records.
*/
public class SalesAssociate
{
    private String name;
    private double sales;

    public SalesAssociate( )
    {
        name = "No record";
        sales = 0;
    }

    public SalesAssociate(String initialName,
                          double initialSales)
    {
        set(initialName, initialSales);
    }
public void set(String newName, double newSales)
   {
       name = newName;
       sales = newSales;
   }

   public void readInput( )
   {
       System.out.print("Enter name of sales associate: ");
       Scanner keyboard = new Scanner(System.in);
       name = keyboard.nextLine( );

       System.out.print("Enter associate's sales: $");
       sales = keyboard.nextDouble( );
   }

   public void writeOutput( )
   {
       System.out.println("Name: " + name);
       System.out.println("Sales: $" + sales);
   }

   public String getName( )
   {
       return name;
   }
   public double getSales( )
   {
       return sales;
   }
}
```

Listing 7.4 Sales Report Program - SalesReporter.java

```
import java.util.Scanner;
/**   Program to generate a sales report.
*/
public class SalesReporter
{
    private double highestSales;
    private double averageSales;
    private SalesAssociate[] team;  //The array object is
                                //created in getData.
    private int numberOfAssociates; //Same as team.length

    /**
     Reads the number of sales associates and data for each one.
    */
    public void getData( )
    {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter number of sales associates:");
        numberOfAssociates = keyboard.nextInt( );
        team = new SalesAssociate[numberOfAssociates + 1];//We won't use
team[0]
for (int i = 1; i <= numberOfAssociates; i++)
        {
            team[i] = new SalesAssociate( );
            System.out.println("Enter data for associate " + i);
            team[i].readInput( );
            System.out.println( );
        }
    }
/**
    Computes the average and highest sales figures.
    Precondition: There is at least one salesAssociate.
   */
   public void computeStats( )
   {
       double nextSales = team[1].getSales( );
       highestSales = nextSales;
       double sum = nextSales;
       for (int i = 2; i <= numberOfAssociates; i++)
       {
            nextSales = team[i].getSales( );
            sum = sum + nextSales;
            if (nextSales > highestSales)
                highestSales = nextSales; //highest sales so far.
       }
       averageSales = sum / numberOfAssociates;
   }
```

An array of a class can be declared and the class's methods applied to the elements of the array.

This excerpt from the Sales Report program in the text uses the `SalesAssociate` class to create an array of sales associates:

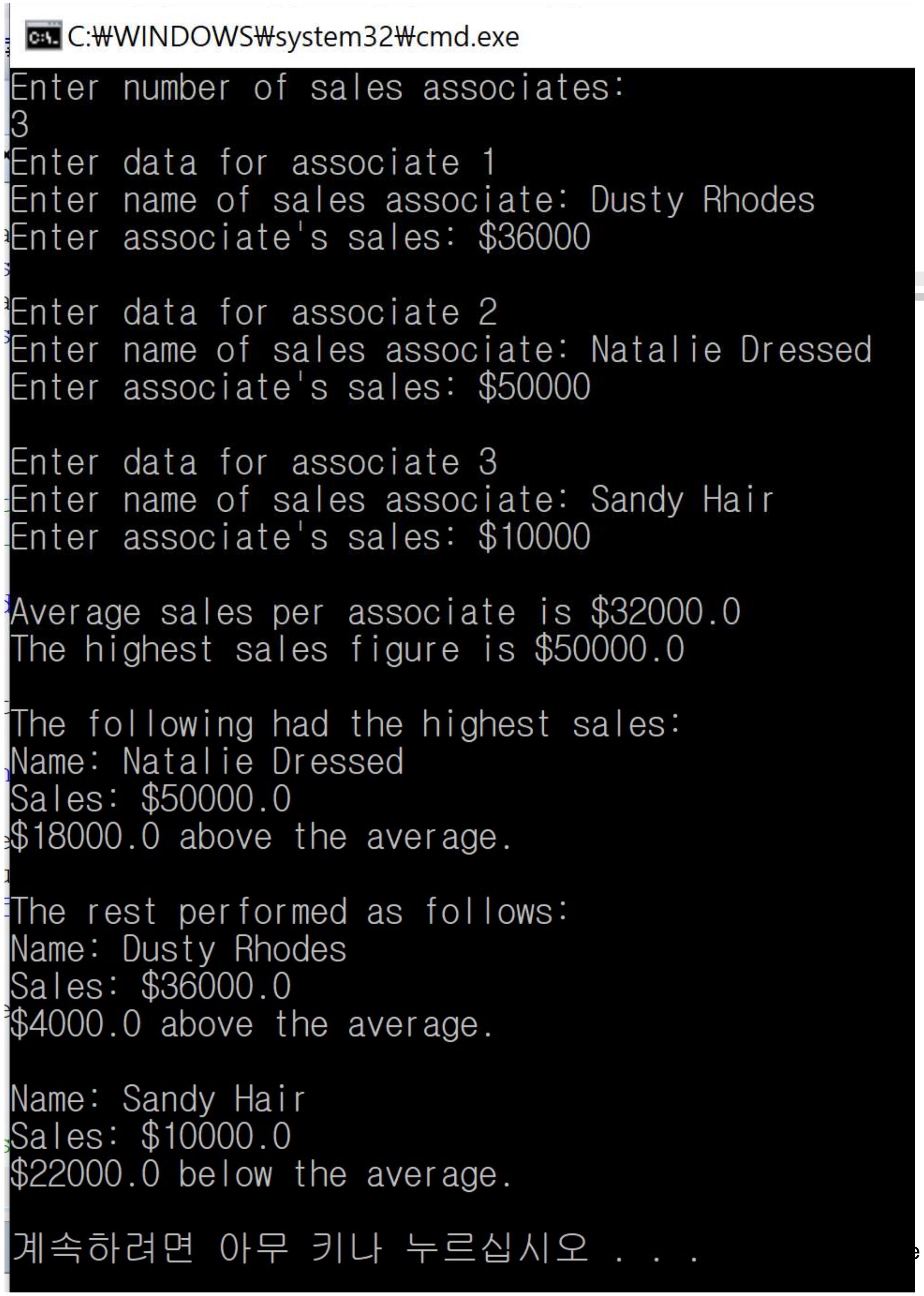
create an array of `SalesAssociate`s

each array element is a `SalesAssociate` instance variable

use the `readInput` method of `SalesAssociate`

```
public void getData( )
   {
      Scanner keyboard = new Scanner(System.in);
      System.out.println("Enter number of sales associates:");
      numberOfAssociates = keyboard.nextInt( );
      team = new SalesAssociate[numberOfAssociates + 1];//We won't
use team[0]
      for (int i = 1; i <= numberOfAssociates; i++)
      {
         team[i] = new SalesAssociate( );
         System.out.println("Enter data for associate " + i);
         team[i].readInput( );
         System.out.println( );
      }
   }
```

```
/**
    Displays sales report on console screen.
   */
   public void displayResults( )
   {
        //……………….
        for (i = 0; i < numberOfAssociates; i++)
        {
            double nextSales = record[i].getSales( );
            if (nextSales == highest)
            {
                team[i].writeOutput( );
                System.out.println("$" + (nextSales - average)
                                       + " above the average.");
                System.out.println( );
            }
        }
        //…………………..
   }

  public static void main(String[] args)
   {
       SalesReporter clerk = new SalesReporter( );
       clerk.getData( );
       clerk.computeStats( );
       clerk.displayResults( );
   }}
```


C:₩WINDOWS₩system32₩cmd.exe

```
Enter number of sales associates:
3
Enter data for associate 1
Enter name of sales associate: Dusty Rhodes
Enter associate's sales: $36000

Enter data for associate 2
Enter name of sales associate: Natalie Dressed
Enter associate's sales: $50000

Enter data for associate 3
Enter name of sales associate: Sandy Hair
Enter associate's sales: $10000

Average sales per associate is $32000.0
The highest sales figure is $50000.0

The following had the highest sales:
Name: Natalie Dressed
Sales: $50000.0
$18000.0 above the average.

The rest performed as follows:
Name: Dusty Rhodes
Sales: $36000.0
$4000.0 above the average.

Name: Sandy Hair
Sales: $10000.0
$22000.0 below the average.

계속하려면 아무 키나 누르십시오 . . .
```

Arrays and Array Elements as Method Arguments

- both an indexed element and an array name can be an argument in a method
- methods can return an array value or an array name

Indexed Variables as Method Arguments

```
public static void main(String arg[])
{
  Scanner keyboard = new Scanner(System.in);
   System.out.println("Enter your score on exam 1:");
   int firstScore = keyboard.nextInt();
   int[ ] nextScore = new int[3];
   int i;
   double possibleAverage;
   for (i = 0; i < nextScore.length; i++)
      nextScore[i] = 80 + 10*i;
   for (i = 0; i < nextScore.length; i++)
   {
      possibleAverage = average(firstScore, nextScore[i]);
      System.out.println("If your score on exam 2 is "
                    + nextScore[i]);
      System.out.println("your average will be "
                    + possibleAverage);
   }
}
public static double average(int n1, int n2)
{
  return (n1 + n2)/2.0;
}
```

`nextScore` is an array of `int`s

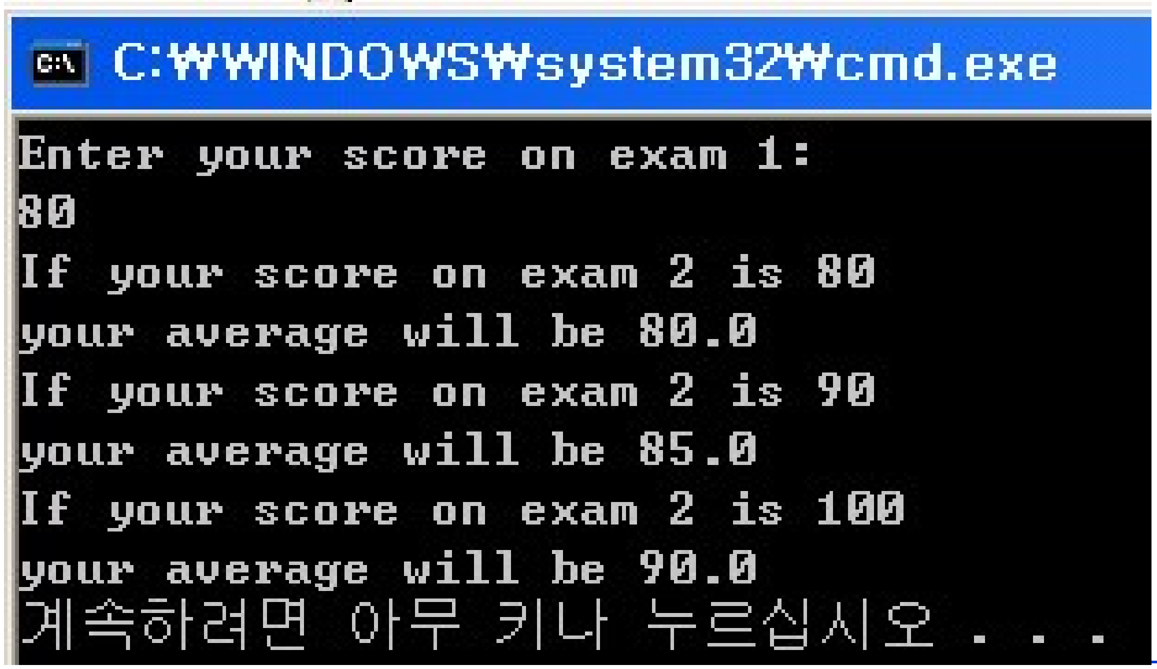
an element of `nextScore` is an argument of method `average`

`average` method definition

Excerpt from ArgumentDemo program in text.

Listing 7.5 Indexed Variables as Arguments –ArgumentDemo.java

```
import java.util.Scanner;

/**
 A demonstration of using indexed variables as arguments.
*/
public class ArgumentDemo
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter your score on exam 1:");
        int firstScore = keyboard.nextInt( );
        int[] nextScore = new int[3];

        for (int i = 0; i < nextScore.length; i++)
            nextScore[i] = firstScore + 5 * i;
for (int i = 0; i < nextScore.length; i++)
        {
            double possibleAverage = getAverage(firstScore, nextScore[i]);
            System.out.println("If your score on exam 2 is " +
                                  nextScore[i]);
            System.out.println("your average will be " +
                                  possibleAverage);
        }
    }

    public static double getAverage(int n1, int n2)
    {
        return (n1 + n2) / 2.0;
    }
}
```



```
Enter your score on exam 1:
80
If your score on exam 2 is 80
your average will be 80.0
If your score on exam 2 is 90
your average will be 85.0
If your score on exam 2 is 100
your average will be 90.0
계속하려면 아무 키나 누르십시오 . . .
```

When Can a Method Change an Indexed Variable Argument?

Ex) doStuff(a[I])

- When a[I] is primitive type(primitive types are **call-by-value)**
  - » only a copy of the value is passed as an argument in a method call
  - » so the method *cannot* change the value of the indexed variable
- When base type of the array a is a class…
  - » class types are reference types; they pass the address of the object when they are an argument in a method call
  - » the corresponding argument in the method definition becomes another name for the object
  - » the method has access to the actual object
  - » so the method *can* change the value of the indexed variable if it is a class (and not a primitive) type

Array Names as Method Arguments

When using **an entire array** as an argument to a method:

- use **just the array name** and no brackets

- the method has access to the original array
- can change the value of the elements

- the length of the array passed can be different for each call
  - » when you define the function you do not know the length of the array that will be passed
  - » so **use** **attribute** inside the method to avoid `ArrayIndexOutOfBoundsException`s

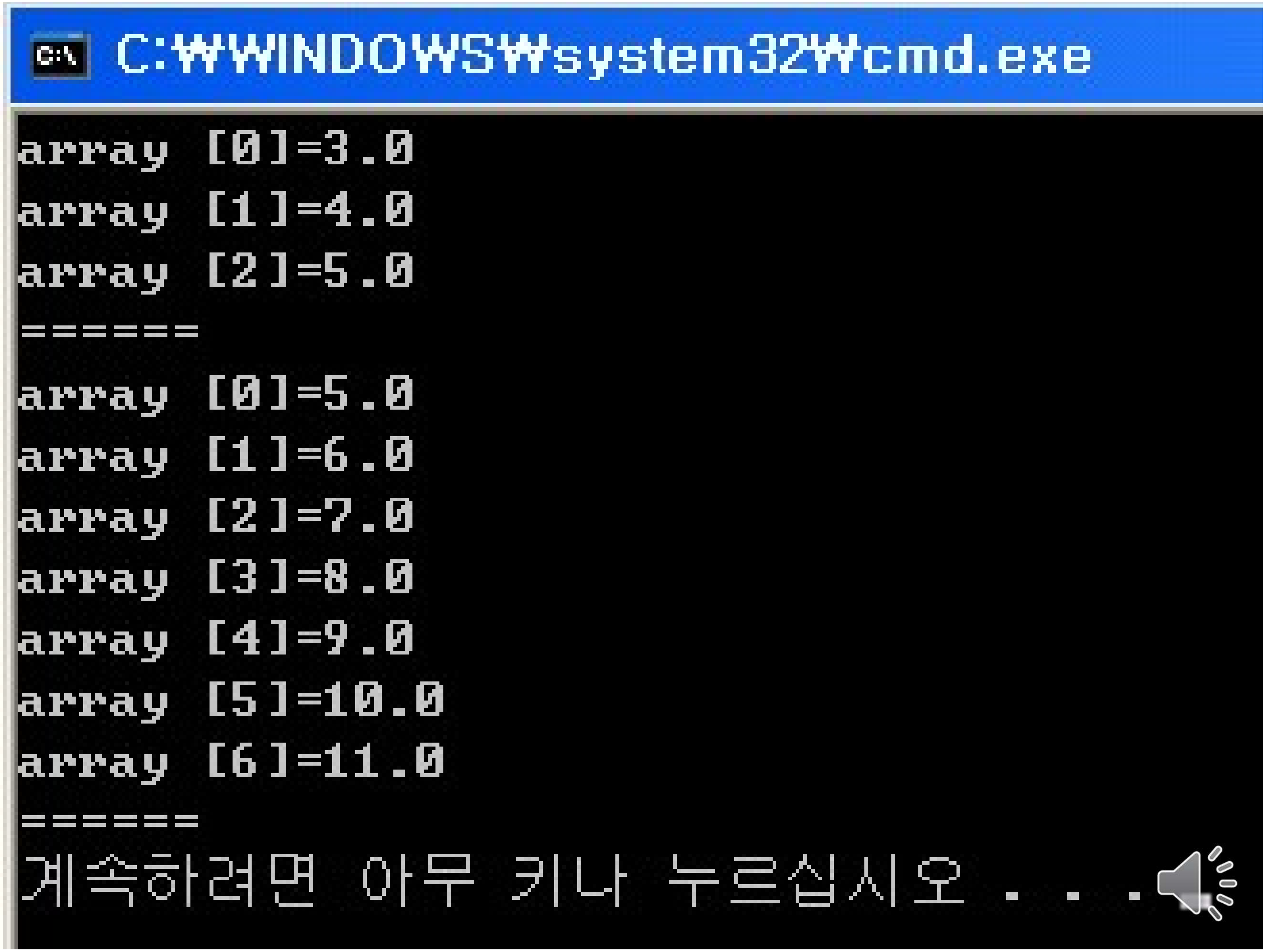
Example: An Array as an Argument in a Method Call

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

the method's argument is the name of an array of characters

uses the `length` attribute to control the loop allows different size arrays and avoids index-out-of-bounds exceptions

```
public class EntireArraysArguments
{
    public static void incrementArrayBy2(double[] a)
   {
                                int i;
                                for (i=0; i<a.length; i++)
                                        a[i] = a[i] + 2;
    }
    public static void showArray(double[] a)
    {
                          int i;
                                for ( i=0; i<a.length; i++)
                                        System.out.println("array ["+i+"]="+a[i]);
                                System.out.println("======");
    }

    public static void main(String[] args)
    {
         double[] a = {1.0, 2.0, 3.0 };
         double[] b = {3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0};
         EntireArraysArguments.incrementArrayBy2(a);
         EntireArraysArguments.incrementArrayBy2(b);
         EntireArraysArguments.showArray(a);
         EntireArraysArguments.showArray(b);
    }
}
```



```
array [0]=3.0
array [1]=4.0
array [2]=5.0
=======
array [0]=5.0
array [1]=6.0
array [2]=7.0
array [3]=8.0
array [4]=9.0
array [5]=10.0
array [6]=11.0
=======
계속하려면 아무 키나 누르십시오 . . .
```

Arguments for the Method **`main`**

- The heading for the `main` method shows a parameter that is an array of `String`s:

  ```
  public static void main(String[] args)
  ```

- When you run a program from the command line, all words after the class name will be passed to the main method in the args array.

  ```
  java TestProgram Josephine Student
  ```

- The following `main` method in the class `TestProgram` will print out the first two arguments it receives:

```
Public static void main(String[] args)
{
   System.out.println("Hello " + args[0] + " " + args[1]);
}
```

- In this example, the output from the command line above will be:

Using = with Array Names: Remember They Are Reference Types

```
int[] a = new int[3];
int[] b = new int[3];
for(int i=0; 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]);
```

This does not create a copy of array `a`; it makes `b` another *name* for array `a`.

The output for this code will be:

A value changed in `a` is the same value obtained with `b`

Using == with array names: remember they are reference types

```
int i;
int[] a = new int[3];
int[] b = new int[3];
for(i; i < a.length; i++)
   a[i] = i;
for(i; 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");
```

`a` and `b` are both 3-element arrays of `int`s

all elements of `a` and `b` are assigned the value `0`

tests if the of `a` and `b` are equal, not if the array values are equal

The output for this code will be " because the *addresses* of the arrays are not equal.

Testing Two Arrays for Equality

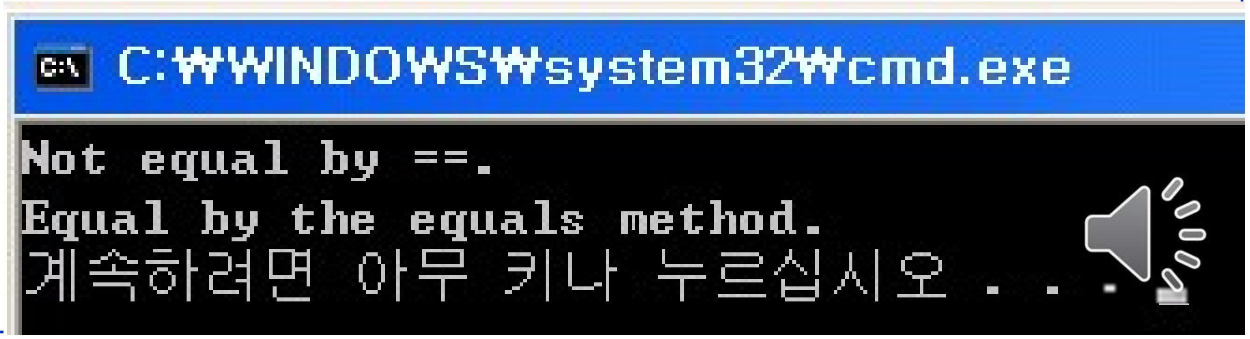
- To test two arrays for equality you need to define an **`equals`** method that returns true if and only the arrays have the same length and all corresponding values are equal

- This code shows an example of an `equals` method.

```
public static boolean equals(int[] a, int[] b)
 {
    boolean match;
    if (a.length != b.length)
       match = false;
    else
    {
       match = true; //tentatively
       int i = 0;
       while (match && (i < a.length))
       {
          if (a[i] != b[i])
             match = 
          i++;
       }
    }
    return match;
 }
```

Listing 7.6 Two Kinds of Equality - TestEquals.java

```
// Listing 7.6 Two Kinds of Equality

/**
 This is just a demonstration program to see how
 equals and == work.
*/
public class TestEquals
{
    public static void main(String[] args)
    {
        int[] a = new int[3]; int[] b = new int[3]; int i;

        // The arrays a and b contain the same integers in the same order
        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("Equal by ==.");
        else
            System.out.println("Not equal by ==.");  //
        if (equals(b,a))
            System.out.println("Equal by the equals method.");  //
        else
            System.out.println("Not equal by the equals method.");
    }
    public static boolean equals(int[] a, int[] b)
    {
        boolean match;
        if (a.length != b.length)
            match = false;
        else
        {
            match = true; //tentatively
            int i = 0;
            while (match && (i < a.length))
            {
                if (a[i] != b[i])
                    match = false;
                i++;
            }
        }
        return match;
    }
}
```



C:₩WINDOWS₩system32₩cmd.exe

```
Not equal by ==.
Equal by the equals method.
계속하려면 아무 키나 누르십시오 . . .
```

Methods that Return an Array

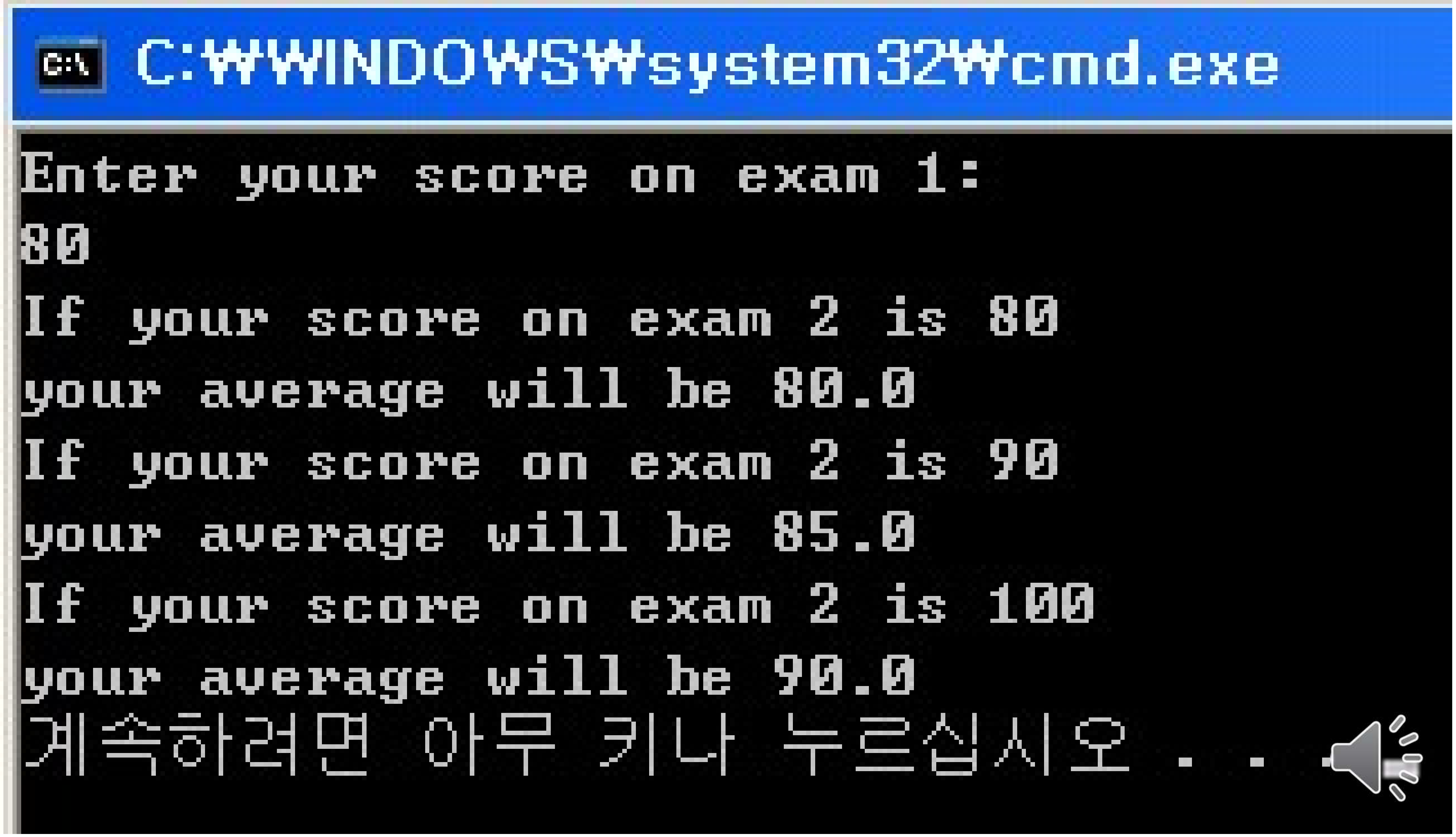
- Yet another example of passing a reference
- Actually, the array is not passed, the **address of the array** is passed
- The local array name within the method is just another name for the original array
- The code at right shows an example of returning an array

```
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;
    }
 }
```

`c`, `newArray`, and the return type of `vowels` are all the same type: `char []`

Listing 7.7A Method that Returns an Array. -ReturnArrayDemo.java

```
import java.util.Scanner;

/**
 A demonstration of a method that returns an array.
*/
public class ReturnArrayDemo
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter your score on exam 1:");
        int firstScore = keyboard.nextInt( );
        int[] nextScore = new int[3];

        for (int i = 0; i < nextScore.length; i++)
            nextScore[i] = firstScore + 5 * i;
        double[] averageScore = getArrayOfAverages(firstScore, nextScore);
        for (int i = 0; i < nextScore.length; i++)
        {
            System.out.println("If your score on exam 2 is " +
                               nextScore[i]);
            System.out.println("your average will be " +
                               averageScore[i]);
        }
    }

    public static double[] getArrayOfAverages(int firstScore, int[] nextScore)
    {
        double[] temp = new double[nextScore.length];
        for (int i = 0; i < temp.length; i++)
            temp[i] = getAverage(firstScore, nextScore[i]);

        return temp;
    }

    public static double getAverage(int n1, int n2)
    {
       return (n1 + n2) / 2.0;
    }
}
```



C:₩WINDOWS₩system32₩cmd.exe

```
Enter your score on exam 1:
80
If your score on exam 2 is 80
your average will be 80.0
If your score on exam 2 is 90
your average will be 85.0
If your score on exam 2 is 100
your average will be 90.0
계속하려면 아무 키나 누르십시오 . . .
```

끝