JAVA ARRAYS

ARRAYS

- Creating and Accessing Arrays **(note that array indices start at 0)**

  BaseType[] ArrayName= **new** BaseType[Length];

- Example:

  double[] temperature = new double[7];

  //collection of seven variables of type double

  - This is like declaring the following seven strangely named variables to have the type double:

temperature[0], temperature[1], temperature[2], temperature[3], temperature[4], temperature[5], temperature[6]

- Since an index can be an expression, a loop can be written to read values into an array, for example:

```
System.out.println("Enter 7 temperatures:");
for(int index = 0;index < 7; index++)
    temperature[index] = keyboard.nextDouble();
```

- Using a named constant for creating an array:

```
public static final int NUMBER_OF_READINGS = 100;
int[] pressure = new int[NUMBER_OF_READINGS];
```

- If you want an array to hold entries, each of which is an object of a class called Species:

```
Species[] entry = new Species[20];
```

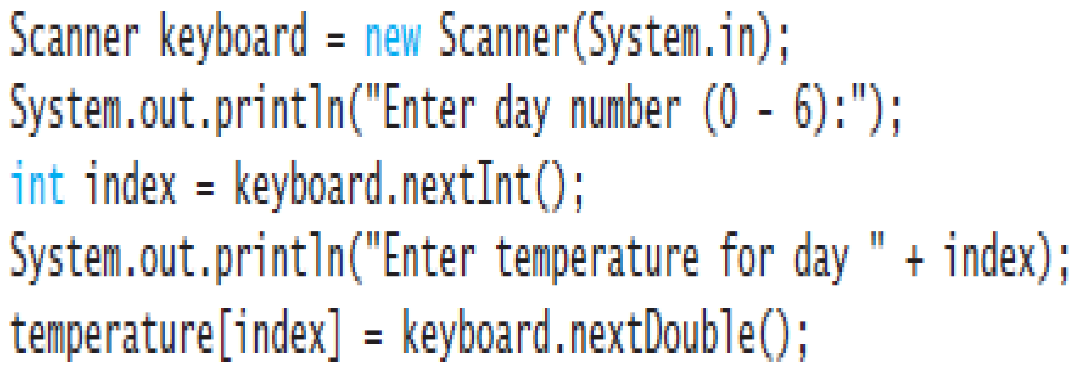
- You can use array elements just like any other variable:

temperature[3] = 32;

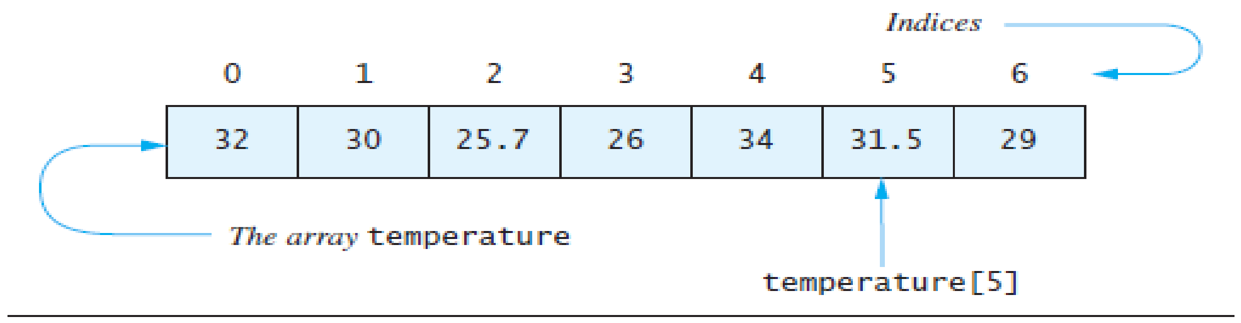
temperature[6] = temperature[3] + 5;

System.out.println(temperature[6]);

- Another example:



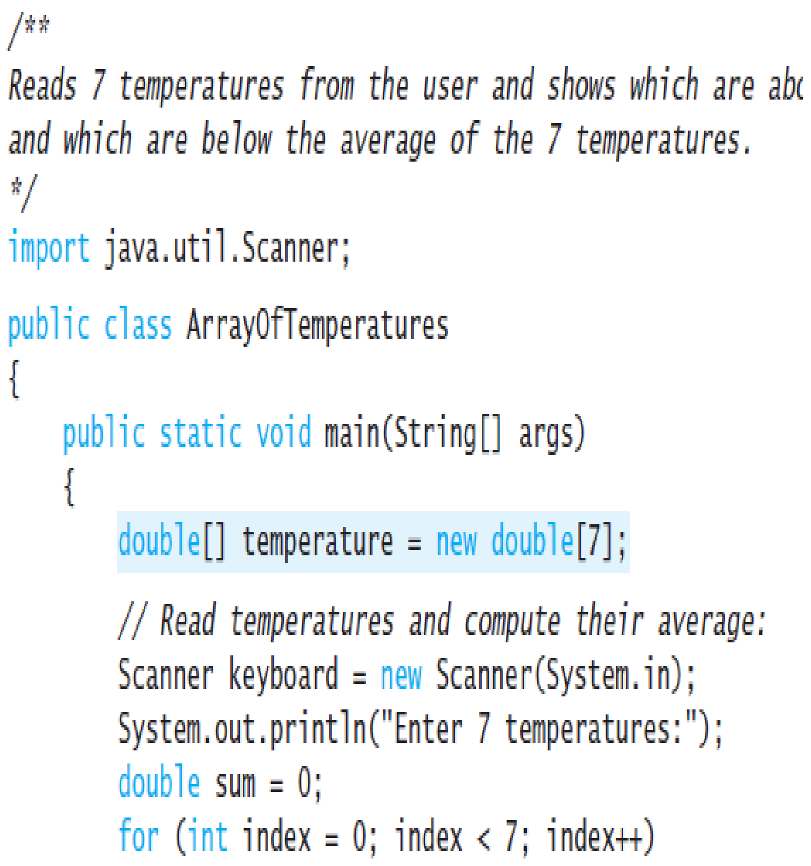
```
Scanner keyboard = new Scanner(System.in);
System.out.println("Enter day number (0 - 6):");
int index = keyboard.nextInt();
System.out.println("Enter temperature for day " + index);
temperature[index] = keyboard.nextDouble();
```



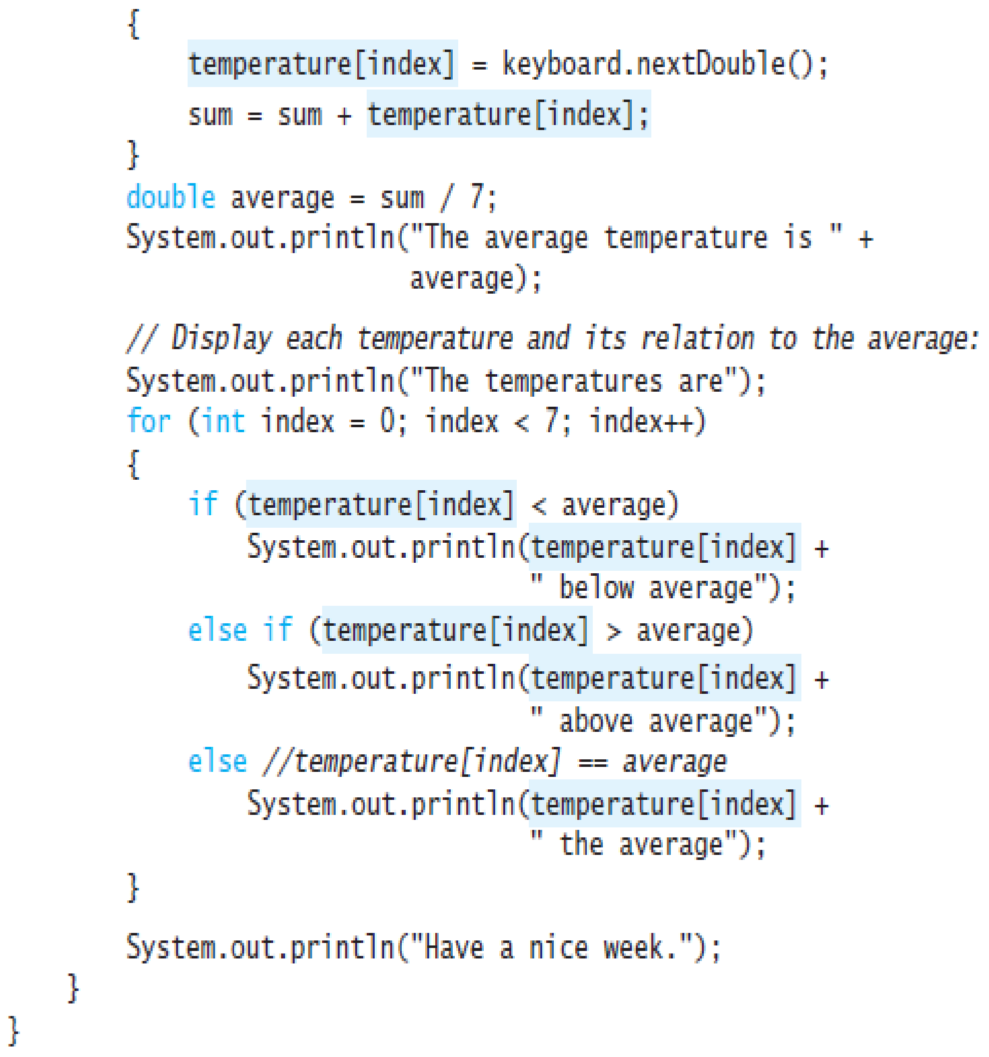
- After the array values are read in, we can display them as follows:

```
System.out.println("The 7 temperatures are:");
for (int index = 0; index < 7; index++)
    System.out.print(temperature[index] + " ");
System.out.println( );
```

- **An Array of Temperatures program**



```
/**
Reads 7 temperatures from the user and shows which are ab
and which are below the average of the 7 temperatures.
*/
import java.util.Scanner;

public class ArrayOfTemperatures
{
    public static void main(String[] args)
    {
        double[] temperature = new double[7];

        // Read temperatures and compute their average:
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter 7 temperatures:");
        double sum = 0;
        for (int index = 0; index < 7; index++)
```



```
        {
            temperature[index] = keyboard.nextDouble();
            sum = sum + temperature[index];
        }
        double average = sum / 7;
        System.out.println("The average temperature is " +
                           average);

        // Display each temperature and its relation to the average:
        System.out.println("The temperatures are");
        for (int index = 0; index < 7; index++)
        {
            if (temperature[index] < average)
                System.out.println(temperature[index] +
                                   " below average");
            else if (temperature[index] > average)
                System.out.println(temperature[index] +
                                   " above average");
            else //temperature[index] == average
                System.out.println(temperature[index] +
                                   " the average");
        }

        System.out.println("Have a nice week.");
    }
}
```

- Sample output, try different numbers…

Sample Screen Output

```
Enter 7 temperatures:
32
30
25.7
26
34
31.5
29
The average temperature is 29.7428
The temperatures are
32.0 above average
30.0 above average
25.7 below average
26.0 below average
34.0 above average
31.5 above average
29.0 below average
Have a nice week.
```

Some details:

- The following creates an array named **pressure**, that is equivalent to 100 variables of type int

```
int[] pressure = new int[100];
```

- Alternatively, you can do the above in 2 steps:

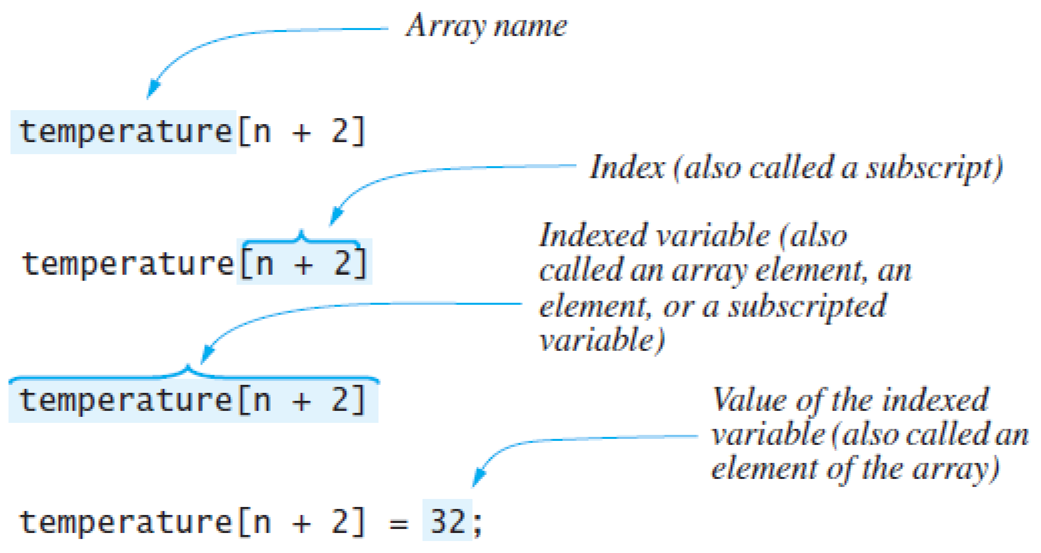
```
int[] pressure;
pressure = new int[100];
```

- The first step declares the variable as an array of integers.
- The second step allocates enough memory for holding 100 integers. (Base type is :   **int**)

- Base type of an array can be any data type.
- Base type can be a class type too… The following statement creates an array named ***entry*** whose elements are Species objects…. (Species is a class).

```
Species[] entry = new Species[3];
```

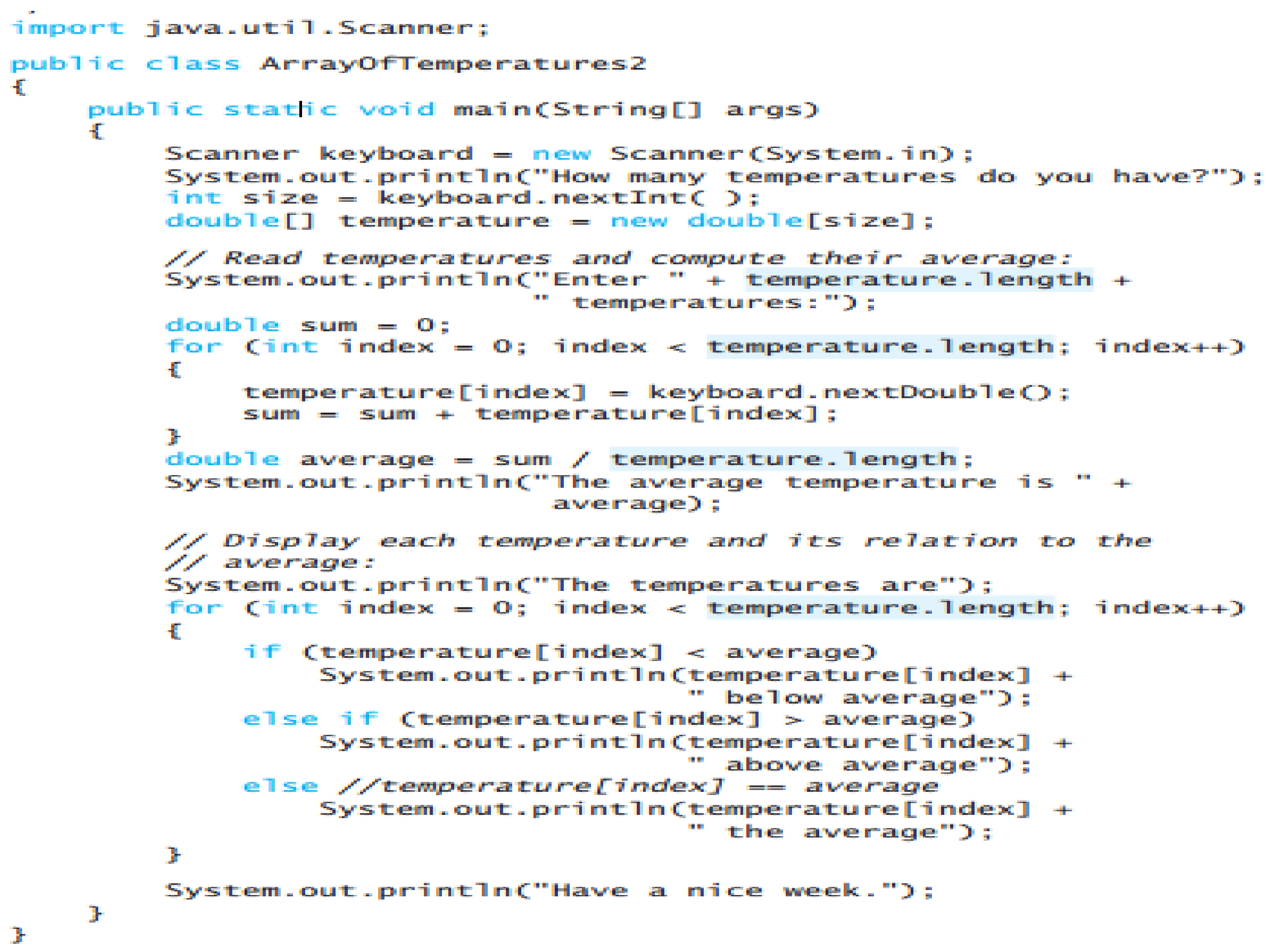
- Array Terminology


Array name
temperature[n + 2]
Index (also called a subscript)
temperature[n + 2]
Indexed variable (also called an array element, an element, or a subscripted variable)
temperature[n + 2]
Value of the indexed variable (also called an element of the array)
temperature[n + 2] = 32;

- An array has only one public instance variable, namely the variable length

example : **entry.length**

Array of temperatures , example code – rewritten using .length() method:



```
import java.util.Scanner;
public class ArrayOfTemperatures2
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("How many temperatures do you have?");
        int size = keyboard.nextInt( );
        double[] temperature = new double[size];

        // Read temperatures and compute their average:
        System.out.println("Enter " + temperature.length +
                           " temperatures:");
        double sum = 0;
        for (int index = 0; index < temperature.length; index++)
        {
            temperature[index] = keyboard.nextDouble();
            sum = sum + temperature[index];
        }
        double average = sum / temperature.length;
        System.out.println("The average temperature is " +
                           average);

        // Display each temperature and its relation to the
        // average:
        System.out.println("The temperatures are");
        for (int index = 0; index < temperature.length; index++)
        {
            if (temperature[index] < average)
                System.out.println(temperature[index] +
                                   " below average");
            else if (temperature[index] > average)
                System.out.println(temperature[index] +
                                   " above average");
            else //temperature[index] == average
                System.out.println(temperature[index] +
                                   " the average");
        }

        System.out.println("Have a nice week.");
    }
}
```

- **Sample Screen Output**

```
How many temperatures do you have?
3
Enter 3 temperatures:
32
26.5
27
The average temperature is 28.5
The temperatures are
32.0 above average
26.5 below average
27.0 below average
Have a nice week.
```

Using a for Loop to Step Through an Array

```
for (int index = 0; index <temperature.length; index++)
{
   temperature[index] = keyboard.nextDouble();
   sum = sum + temperature[index];
}
```

- Another way to step through an entire array—after its elements have been initialized—is to use for-each statement

```
for (int value : temperature)
{
   if (value < average)
       System.out.println(value + " below average.");
   else if (value > average)
       System.out.println(value + " above average.");
   else //value == average
       System.out.println(value + " the average.");
}
```

- Make sure array indices don’t go out of bounds in calculations:

```
System.out.println("Enter a list of nonnegative integers.");
System.out.println("Place a negative integer at the end.");
int[] list = new int[10];
Scanner keyboard = new Scanner(System.in);
int number = keyboard.nextInt();
int i = 0;


while ( (i <list.length) && (number >= 0) )
{
   list[i] = number;
   i++;
   number = keyboard.nextInt();
}
if (number >= 0)
{
   System.out.println("Could not read in all the numbers.");
   System.out.println("Only able to read" + list.length +
                        " numbers.");
}
```

- Initializing arrays at the time of declaration

double[] reading = {3.3, 15.8, 9.7};

- Same as doing:

double[] reading = new double[3];

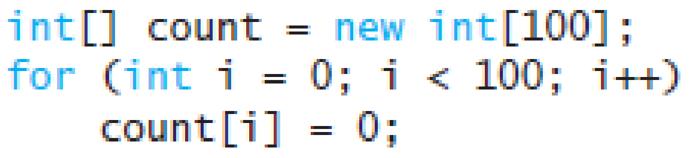
reading[0] = 3.3;

reading[1] = 15.8;

reading[2] = 9.7;

- If you donot initialize an array of int, they get values of 0 automatically…
- Explicit initialization is safer than default initialization.

- You can use a loop like this:



```
int[] count = new int[100];
for (int i = 0; i < 100; i++)
    count[i] = 0;
```

ARRAYS IN CLASSES AND METHODS

- **Arrays can be used as instance variables in classes.**
- **Methods can have an indexed variable or an entire array as an argument, and can return an array.**

- **Next, Sales Report example:**
- **Name and Sales Figures: can design a class for a single sales associate that holds these two data items. Our class can perform input and output and have a reasonable complement of accessor and mutator methods.**

```java
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;
    }
}
```

- We can use the following instance variables to record the desired data:

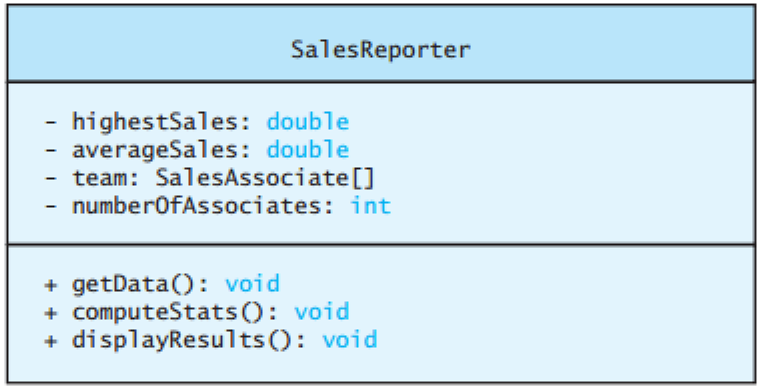
private double highestSales;

private double averageSales;

private SalesAssociate[] team;

- The job of our program breaks down into these main subtasks:

1. Get ready.
2. Obtain the data.
3. Compute some statistics
4. Display the results



- So, to organize our thoughts , the class will look like this

```
public class SalesReporter
{
        private double highestSales;
        private double averageSales;
        private SalesAssociate[] team;
        private int numberOfAssociates; //Same as team.length

        public static void main(String[ ] args)
       {
           SalesReporter clerk = new SalesReporter();
           clerk.getData();
           clerk.computeStats();
           clerk.displayResults();
         }
        <More stuff needs to be added here.>
}
```

```
import java.util.Scanner;
/**
 Program to generate 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];
        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();
        }
    }
```

The `main` *method is at the end of the class.*

Array object created here.

`SalesAssociate` *objects created here.*

```
/**
 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;
}
/**
 Displays sales report on the screen.
*/
public void displayResults()
{
    System.out.println("Average sales per associate is $" +
                        averageSales);
    System.out.println("The highest sales figure is $" +
                        highestSales);
    System.out.println();
    System.out.println("The following had the highest sales:");
    for (int i = 1; i <= numberOfAssociates; i++)
    {
        double nextSales = team[i].getSales();
        if (nextSales == highestSales)
        {
            team[i].writeOutput();
            System.out.println("$" + (nextSales - averageSales)
                              + " above the average.");
            System.out.println();
        }
    }

    System.out.println("The rest performed as follows:");
    for (int i = 1; i <= numberOfAssociates; i++)
    {
        double nextSales = team[i].getSales();
        if (team[i].getSales() != highestSales)
        {
```

Already processed `team[1]`, *so the loop starts with* `team[2]`.

```
                team[i].writeOutput();
                if (nextSales >= averageSales)
                    System.out.println("$" + (nextSales -
                           averageSales) + " above the average.");
                else
                    System.out.println("$" + (averageSales -
                              nextSales) + " below the average.");
                System.out.println();
            }
        }
    }
    public static void main(String[] args)
    {
        SalesReporter clerk = new SalesReporter();
        clerk.getData();
        clerk.computeStats();
        clerk.displayResults();
    }
}
```

Sample Screen Output

```
Enter number of sales associates:
3
Enter data for associate number 1
Enter name of sales associate: Dusty Rhodes
Enter associate's sales: $36000
Enter data for associate number 2
Enter name of sales associate: Natalie Dressed
Enter associate's sales: $50000
Enter data for associate number 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.
```

Indexed Variables as Method Arguments

- An indexed variable can be used anywhere as a regular argument. Examples:

```
double possibleAverage = getAverage(firstScore,nextScore[i]);
```

```
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;
    }
}
```

Sample Screen Output

```
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 85
your average will be 82.5
If your score on exam 2 is 90
your average will be 85.0
```

Entire Arrays as Arguments to a Method

- A parameter can represent an entire array

```
public class SampleClass
{
    public static void incrementArrayBy2(double[] anArray)
    {
        for (int i = 0; i <anArray.length; i++)
            anArray[i] = anArray[i] + 2;
    }
    <The rest of the class definition goes here.>
}
```

suppose you have the statements

```
double[] a = new double[10];
double[] b = new double[30];
```

- SampleClass.incrementArrayBy2(a);
- SampleClass.incrementArrayBy2(b);

EXAMPLES

```
public static int getOneElement(char[] anArray, int index)
public void readArray(int[] anotherArray)
```

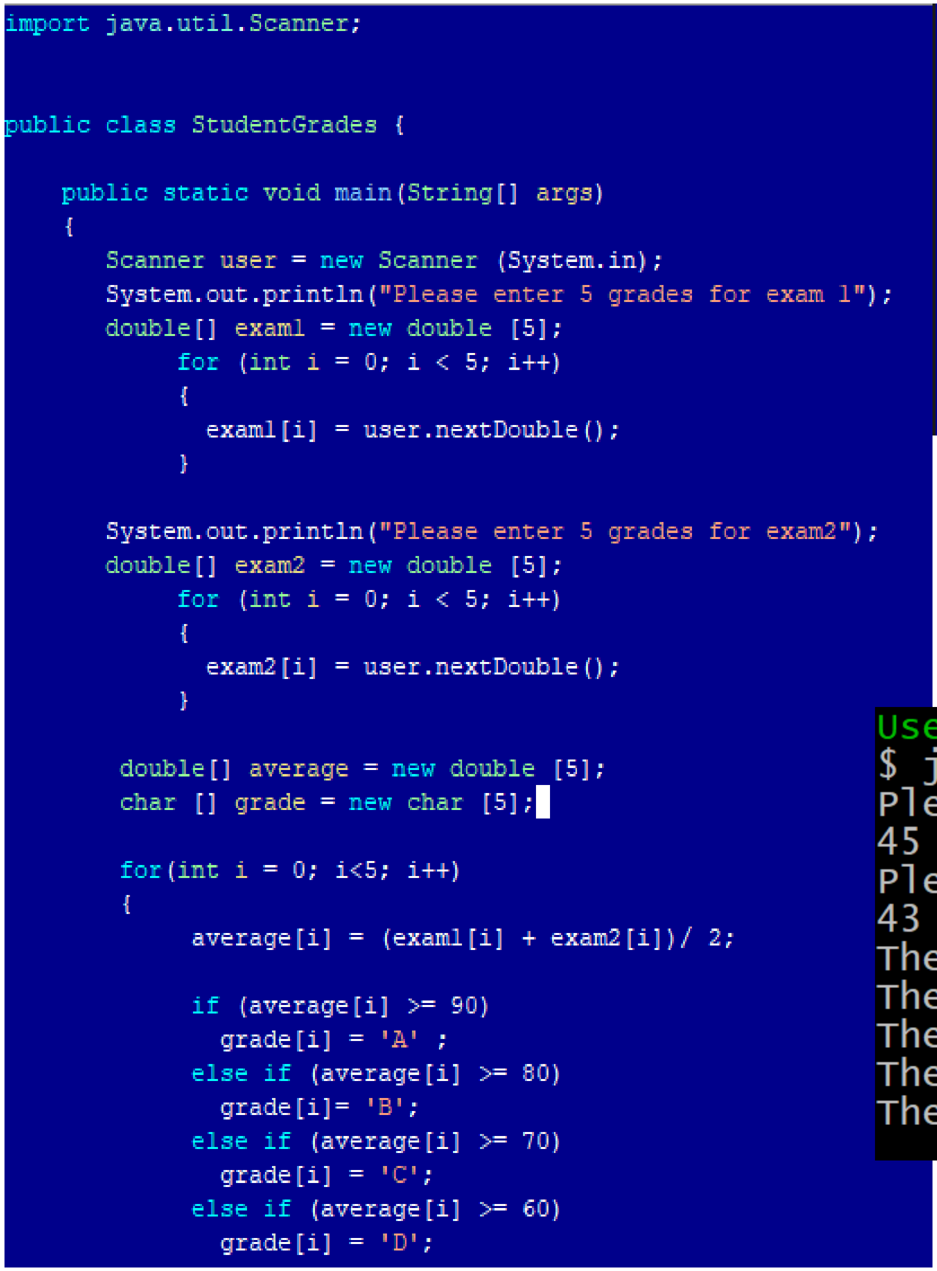
REMEMBER Characteristics of Array Arguments

- No square brackets are written when you pass an entire array as an argument to a method.
- An array of any length can be the argument corresponding to an array parameter.
- A method can change the values in an array argument.

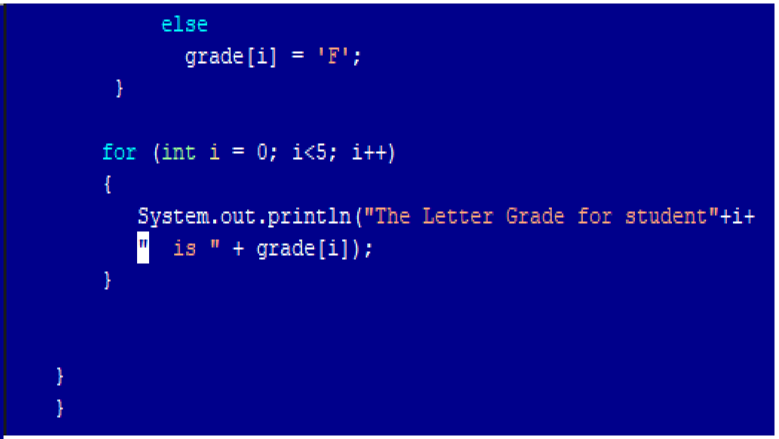
Each of these points is demonstrated by the preceding method `incrementArrayBy2`.

Inclass work

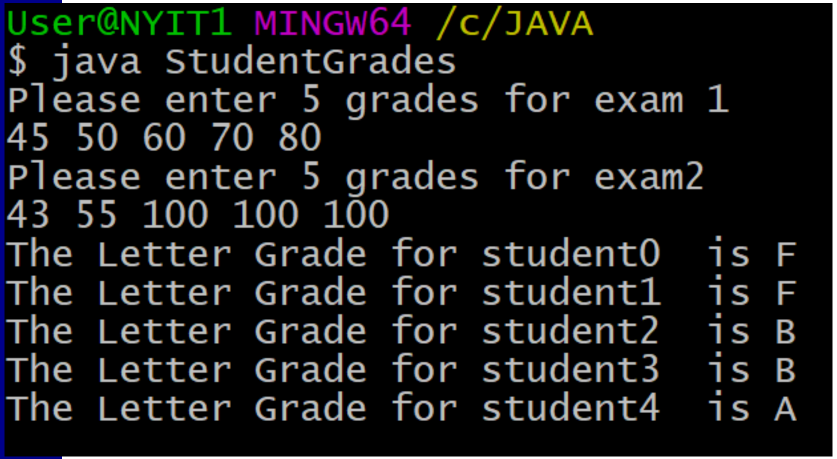
- Create 2 arrays with 5 elements: quiz1 and quiz2
- Then, create getAverage method to calculate averages and to assign a Letter grade for each
- Finally, assign letter a grades to a 5-element character array



```java
import java.util.Scanner;


public class StudentGrades {

    public static void main(String[] args)
    {
       Scanner user = new Scanner (System.in);
       System.out.println("Please enter 5 grades for exam 1");
       double[] exam1 = new double [5];
            for (int i = 0; i < 5; i++)
            {
              exam1[i] = user.nextDouble();
            }

       System.out.println("Please enter 5 grades for exam2");
       double[] exam2 = new double [5];
            for (int i = 0; i < 5; i++)
            {
              exam2[i] = user.nextDouble();
            }

        double[] average = new double [5];
        char [] grade = new char [5];

        for(int i = 0; i<5; i++)
        {
             average[i] = (exam1[i] + exam2[i])/ 2;

             if (average[i] >= 90)
               grade[i] = 'A' ;
             else if (average[i] >= 80)
               grade[i]= 'B';
             else if (average[i] >= 70)
               grade[i] = 'C';
             else if (average[i] >= 60)
               grade[i] = 'D';
```

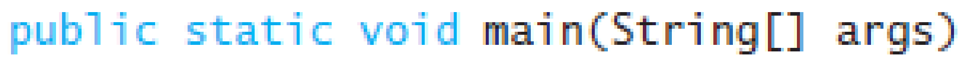


```java
             else
               grade[i] = 'F';
        }

        for (int i = 0; i<5; i++)
        {
          System.out.println("The Letter Grade for student"+i+
          "  is " + grade[i]);
        }


    }
}
```



```
User@NYIT1 MINGW64 /c/JAVA
$ java StudentGrades
Please enter 5 grades for exam 1
45 50 60 70 80
Please enter 5 grades for exam2
43 55 100 100 100
The Letter Grade for student0  is F
The Letter Grade for student1  is F
The Letter Grade for student2  is B
The Letter Grade for student3  is B
The Letter Grade for student4  is A
```

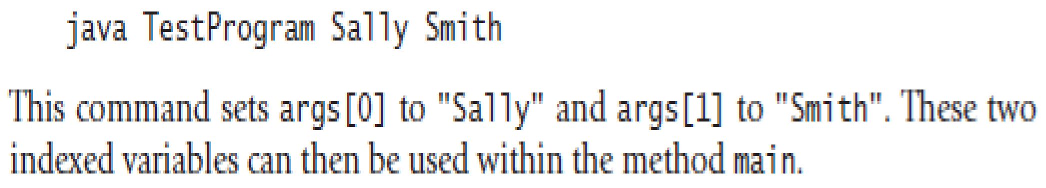
Arguments for the Method main



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

- The parameter declaration String[] args indicates that args is an array whose base type is String

- You can provide additional strings if you like:



```
java TestProgram Sally Smith
```

This command sets args[0] to "Sally" and args[1] to "Smith". These two indexed variables can then be used within the method main.

For example, consider the following sample program:

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

After running TestProgram using the one-line command

```
java TestProgram Josephine Student
```

the output produced by the program will be

```
Hello Josephine Student
```

Array Assignment and Equality

- Cannot use equality operator to compare arrays !

```
/**
 A demonstration program to test two arrays for equality.
*/
public class TestEquals
{
    public static void main(String[] args)
    {
        int[] a = new int[3];
        int[] b = new int[3];
        setArray(a);
        setArray(b);

        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 elementsMatch = true;//tentatively
        if (a.length != b.length)
            elementsMatch = false;
        else
        {
            int i = 0;
            while (elementsMatch && (i < a.length))
            {
                if (a[i] != b[i])
                    elementsMatch = false;
                i++;
            }
         }
         return elementsMatch;
    }

    public static void setArray(int[] array)
    {
         for (int i = 0; i < array.length; i++)
             array[i] = i;
    }
}
```

The arrays a *and* b *contain the same integers in the same order.*

Screen Output

```
Not equal by ==.
Equal by the equals method.
```

- **You cannot use the assignment operator = to give an array more than one name.**

- **You cannot use it to copy the contents of one array to another, different array.**

- **Similarly, the equality operator == tests whether two array names reference the same memory address. It does not test whether two different arrays contain the same values.**

- **To copy elements of one array into a new one:**

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

instead of the assignment statement

```
b = a;
```

Methods That Return Arrays

- A Java method may return an array. To do so, specify the method's return type in the same way you specify the type of an array parameter.

- Example:

```
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;
}
```

- **SYNTAX**

```
public static Base_Type[]  Method_Name(Parameter_List)
{
    Base_Type[] temp = new Base_Type[Array_Size];
    Statements_To_Fill_Array
    return temp;
}
```

The method need not be static and need not be public. The following are some of the other acceptable method headings:

public Base_Type[] Method_Name(Parameter_List)

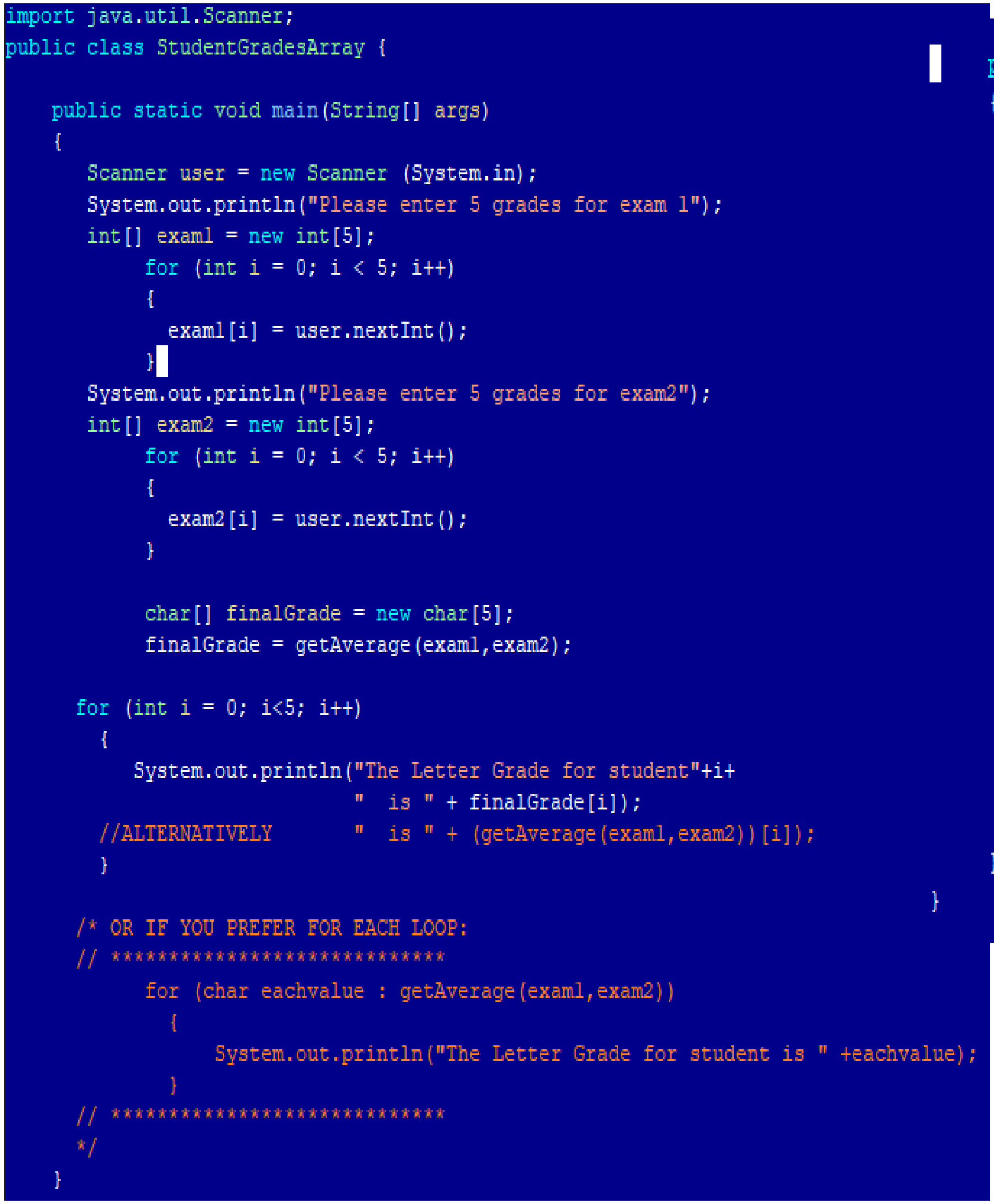
private static Base_Type[ ] Method_Name(Parameter_List)

private Base_Type[] Method_Name(Parameter_List)

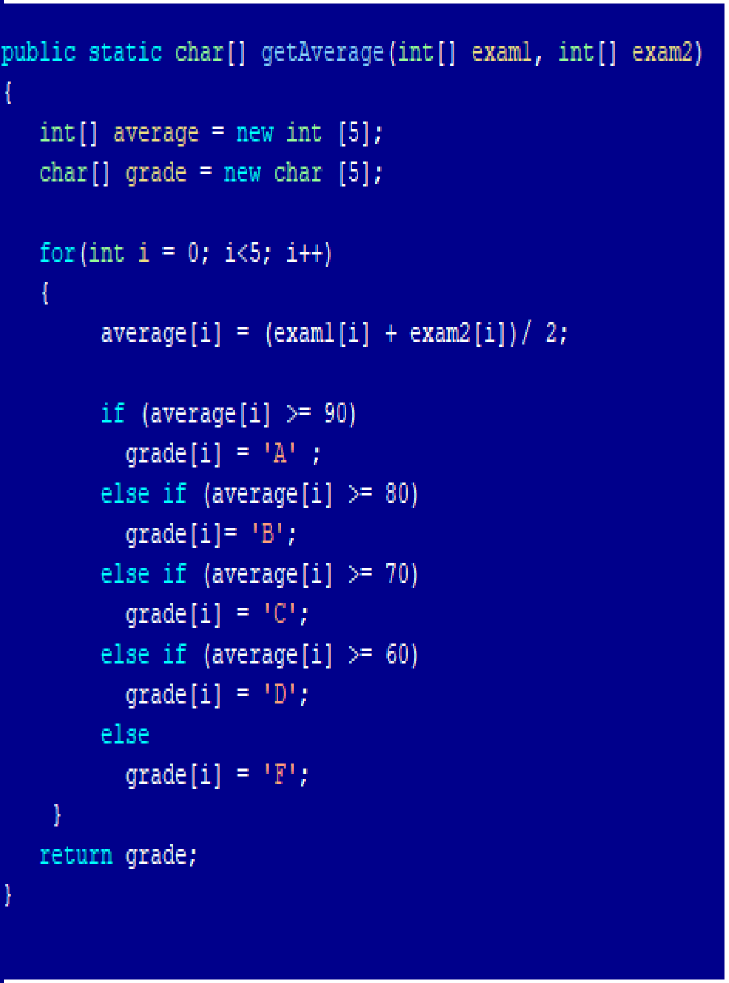
- **EXAMPLE**

```
public static char[] getVowels()
{
char[] newArray = {'a', 'e', 'i', 'o', 'u'};
return newArray;
}
```

Next slide shows an example using our previous inclass exercise……



```java
import java.util.Scanner;
public class StudentGradesArray {

    public static void main(String[] args)
    {
       Scanner user = new Scanner (System.in);
       System.out.println("Please enter 5 grades for exam 1");
       int[] exam1 = new int[5];
            for (int i = 0; i < 5; i++)
            {
              exam1[i] = user.nextInt();
            }
       System.out.println("Please enter 5 grades for exam2");
       int[] exam2 = new int[5];
            for (int i = 0; i < 5; i++)
            {
              exam2[i] = user.nextInt();
            }

            char[] finalGrade = new char[5];
            finalGrade = getAverage(exam1,exam2);

      for (int i = 0; i<5; i++)
        {
           System.out.println("The Letter Grade for student"+i+
                             "  is " + finalGrade[i]);
        //ALTERNATIVELY       "  is " + (getAverage(exam1,exam2))[i]);
        }
                                                                        }

      /* OR IF YOU PREFER FOR EACH LOOP:
      // ******************************
            for (char eachvalue : getAverage(exam1,exam2))
              {
                  System.out.println("The Letter Grade for student is " +eachvalue);
              }
      // ******************************
      */
    }
```



```java
public static char[] getAverage(int[] exam1, int[] exam2)
{
   int[] average = new int [5];
   char[] grade = new char [5];

   for(int i = 0; i<5; i++)
   {
        average[i] = (exam1[i] + exam2[i])/ 2;

        if (average[i] >= 90)
          grade[i] = 'A' ;
        else if (average[i] >= 80)
          grade[i]= 'B';
        else if (average[i] >= 70)
          grade[i] = 'C';
        else if (average[i] >= 60)
          grade[i] = 'D';
        else
          grade[i] = 'F';
    }
   return grade;
}
```

1. What output will be produced by the following code?

```
int[] anArray = new int[10];
for (int i = 0; i <anArray.length; i++)
    anArray[i] = 2 * i;
for (int element : anArray)
    System.out.print(element + " ");
System.out.println();
```

2. What output will be produced by the following code?

```
char[] vowel = {'a', 'e', 'i', 'o', 'u'};
for (int index = 0; index <vowel.length; index++)
    System.out.println(vowel[index]);
```

3. What output will be produced by the following code?

```
double tide[] = {12.2, -7.3, 14.2, 11.3};
System.out.println("Tide 1 is " + tide[1]);
System.out.println("Tide 2 is " + tide[2]);
```

4. Consider the following array:

```
int[] a = new int[10];
```

What is the last index of a? What is the value of `a.length`?

5. What is wrong with the following code to initialize an array b?

```
int[] b = new int[10];
for (int i = 1; i <= b.length; i++)
    b[i] = 5 * i;
```

1.
```
0 2 4 6 8 10 12 14 16 18
```

2.
```
a
e
i
o
u
```

3.
```
Tide 1 is -7.3
Tide 2 is 14.2
```

4. The last index of `a` is 9. The value of `a.length` is 10.

5. The `for` loop references elements `b[1]` through `b[10]`, but there is no element indexed by 10. The array elements are `b[0]` through `b[9]`. If included in a complete class or program, the code would compile without any error messages, but when it is run, you would get an error message saying that an array index is out of bounds.

7. Write some Java code that will declare an array named `entry` that has length 3, has `SalesAssociate` (Listing 7.3) as its base type, and is filled with three identical records. The records use the name `"Jane Doe"` and sales of $5000. Use a `for` loop.

8. Rewrite the method `displayResults` of the program `SalesReporter` (Listing 7.4) so that it uses the methods in the class `DollarFormat` (Listing 6.14 of Chapter 6) to display the dollar amounts in the correct format for dollars and cents.

```
7. SalesAssociate[] entry = new SalesAssociate[3];
   for (int i = 0; i <entry.length; i++)
       entry[i] = new SalesAssociate("Jane Doe", 5000);
```

9. What output will be produced by the following code?

```
char[] a = new char[3];
for (int i = 0; i <a.length; i++)
    a[i] = a;
char[] b = a;
System.out.println("a[1] = " + a[1] + ", b[1] = " + b[1]);
System.out.println("a[2] = " + a[2] + ", b[2] = " + b[2]);
b[2] = b;
System.out.println("a[1] = " + a[1] + ", b[1] = " + b[1]);
System.out.println("a[2] = " + a[2] + ", b[2] = " + b[2]);
```

10. Give the definition of a static method called `showArray` that has an array of base type `char` as a single parameter and that writes one line of text to the screen consisting of the characters in the array argument written in order.

11. Give the definition of a static method called `getArrayOfHalves` that has an array of base type `double` as a single parameter and that returns another array whose base type and length are the same as those of the parameter, but whose elements have each been divided by 2.0.

9.
```
a[1] = a, b[1] = a
a[2] = a, b[2] = a
a[1] = a, b[1] = a
a[2] = b, b[2] = b
```

10. Here are two possible answers:

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

public static void showArray(char[] line)
{
    for (char character : line)
        System.out.print(character);
    System.out.println();
}
```

11.
```
public static double[] getArrayOfHalves(double[] a)
{
    double[] temp = new double[a.length];
    for (int i = 0; i < a.length; i++)
        temp[i] = a[i] / 2.0;
    return temp;
}
```

12. The following method compiles and executes but does not work as you might hope. What is wrong with it?

```
/** Copies an array. */
public static int[] copyArray(int[] anArray)
{
    int[] temp = new int[anArray.length];
    temp = anArray;
    return temp;
}
```

13. The following method compiles and executes but does not work as you might hope. What is wrong with it?

```
/** Doubles the size of an array. */
public static void doubleSize(int[] a)
{
    a = new int[a.length * 2];
}
```

14. Suppose that we add the following method to the class `SalesReporter` in Listing 7.4 so that a program using this class can access the sales associates:

```
/** Returns an array of SalesAssociate objects. */
public SalesAssociate[] getSalesTeam()
{
    return team;
}
```

Will this method compile and execute as indicated? Is adding this method to the class `SalesReporter` a good idea?

12. The method does not return an array distinct from the given argument array. Rather, it returns a reference to the array it is given. To make a duplicate array, you would replace the statement

```
temp = anArray;
```

with

```
for (int i = 0; i <anArray.length; i++)
    temp[i] = anArray[i];
```

13. If b is an array of length 10, the invocation

```
doubleSize(b);
```

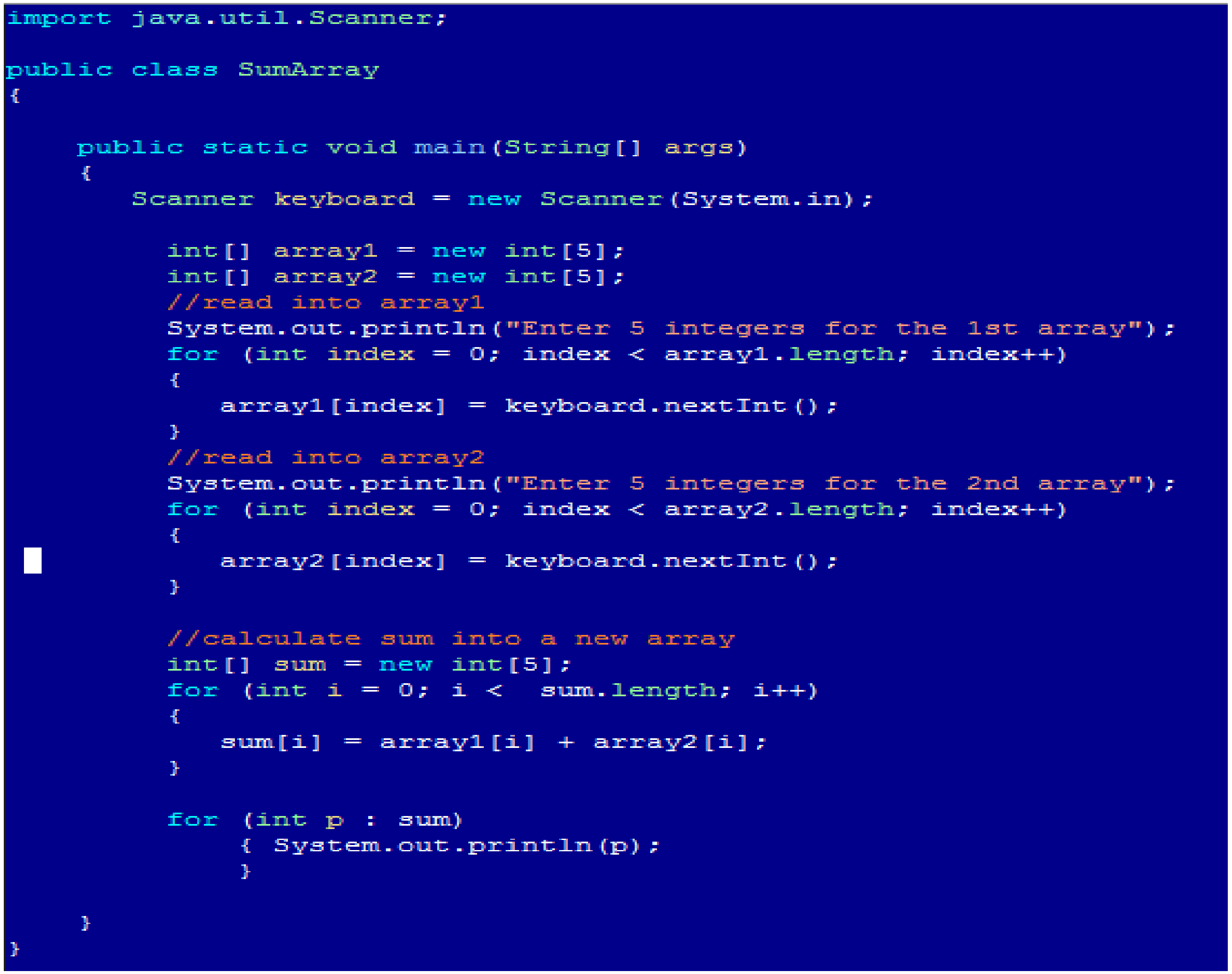
will run with no error messages, but the length of b will not change. In fact, nothing about b will change. The parameter a is a local variable that is initialized with a reference to b. The local variable a is changed so that it contains a reference to an array that is twice the size of b, but that reference goes away as soon as the invocation ends.

14. The method compiles and executes correctly. It returns a reference to the instance variable team, which is an array of SalesAssociate objects. However, this method causes a privacy leak. Once you have a reference to the array team, you could change the data in the array. For example, you could change your competitor's sales to zero by using the SalesAssociate method set.

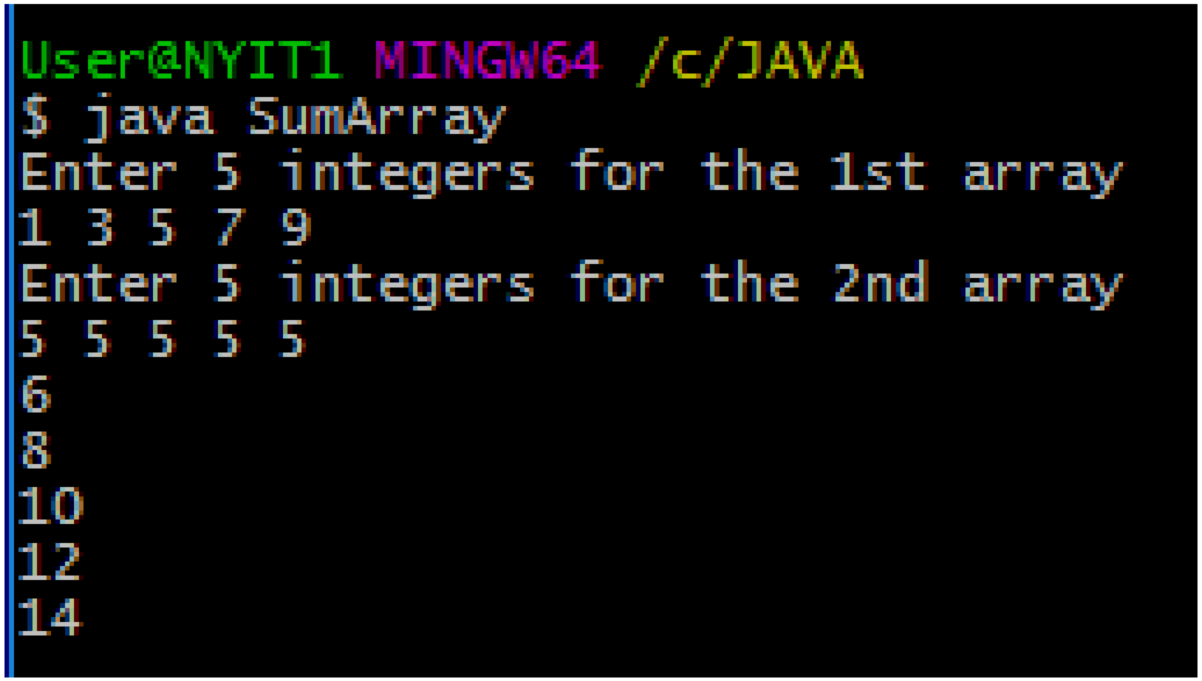
- **InClass Extra credit work1**:

- add elements of 2 int arrays and display results

- array1 is int array of 5 elements
- array2 is int array of 5 elements
- sum is int array of 5 elements to hold sum values



```java
import java.util.Scanner;

public class SumArray
{

    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);

            int[] array1 = new int[5];
            int[] array2 = new int[5];
            //read into array1
            System.out.println("Enter 5 integers for the 1st array");
            for (int index = 0; index < array1.length; index++)
            {
                array1[index] = keyboard.nextInt();
            }
            //read into array2
            System.out.println("Enter 5 integers for the 2nd array");
            for (int index = 0; index < array2.length; index++)
            {
                array2[index] = keyboard.nextInt();
            }

            //calculate sum into a new array
            int[] sum = new int[5];
            for (int i = 0; i <  sum.length; i++)
            {
                sum[i] = array1[i] + array2[i];
            }

            for (int p : sum)
                { System.out.println(p);
                }

    }
}
```



```
User@NYIT1 MINGW64 /c/JAVA
$ java SumArray
Enter 5 integers for the 1st array
1 3 5 7 9
Enter 5 integers for the 2nd array
5 5 5 5 5
6
8
10
12
14
```

- **InClass Extra credit work2**:

- Create a class called **Lib**, for library books, using

  private String title;

  private int id;

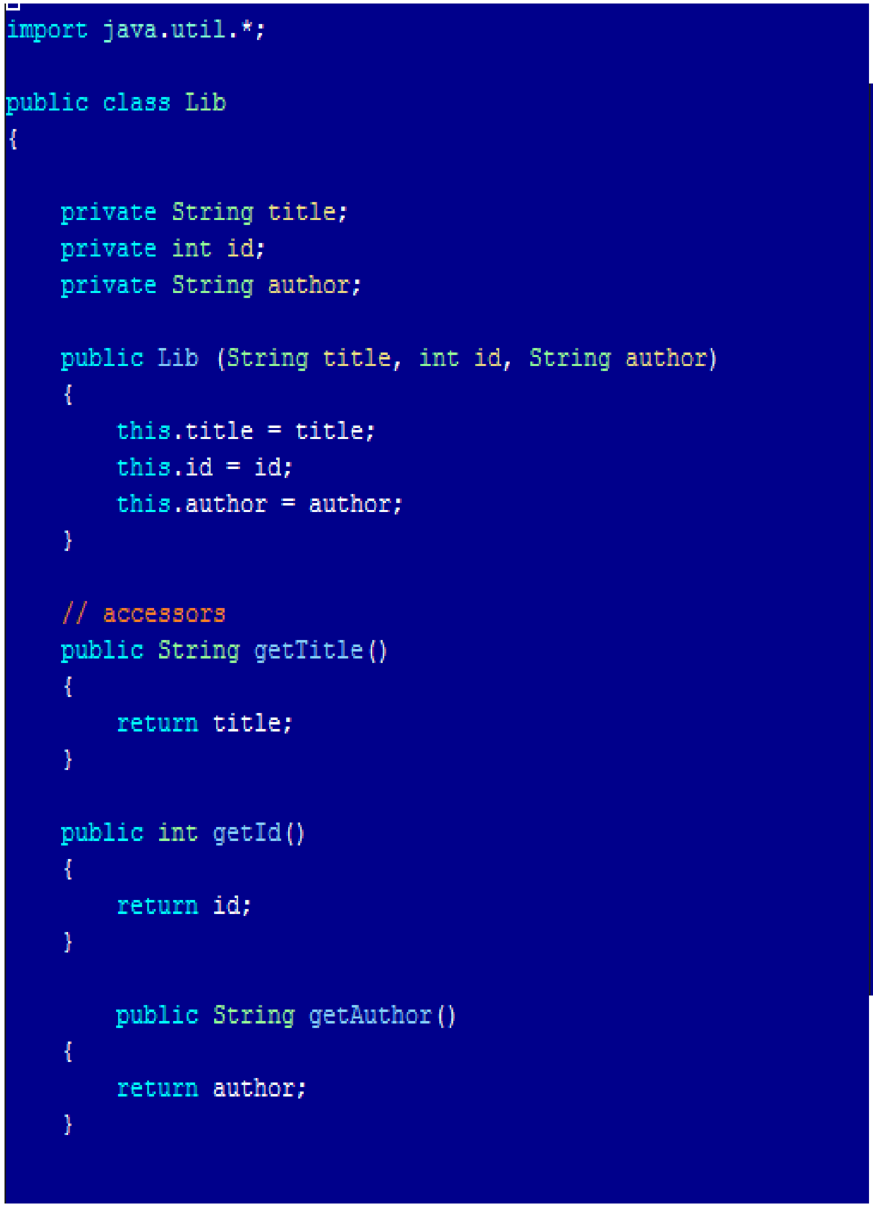
  private String author;

- Then, create a LibraryArrayDemo.java to create 10 books

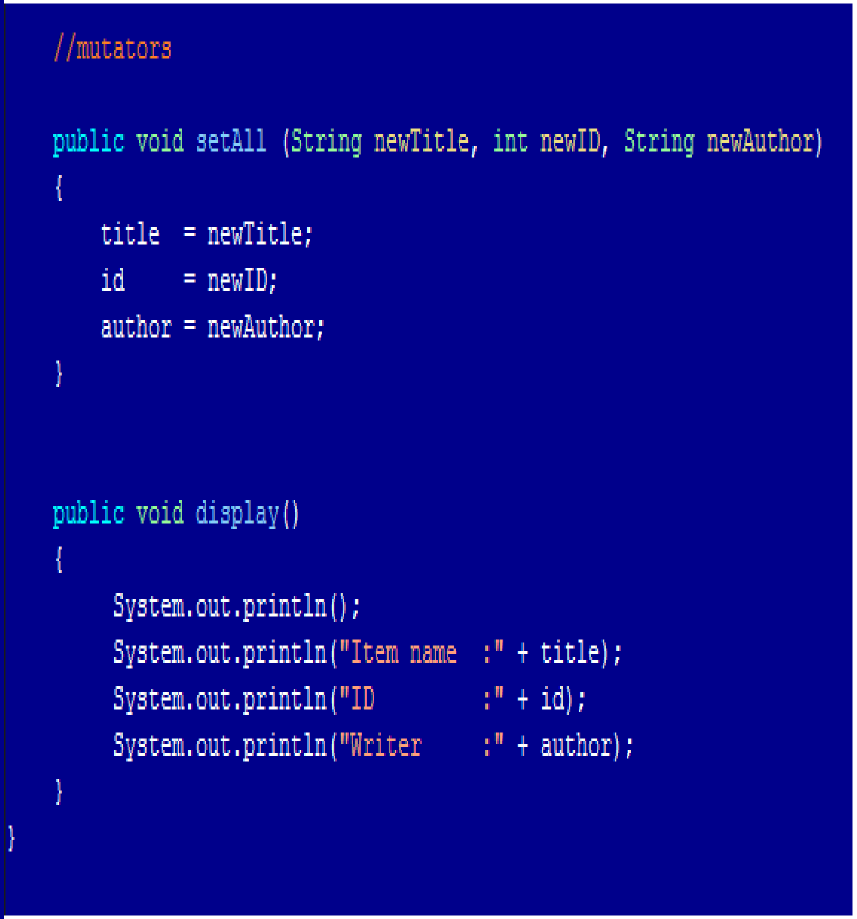
With title = ABC,

author = John Quite

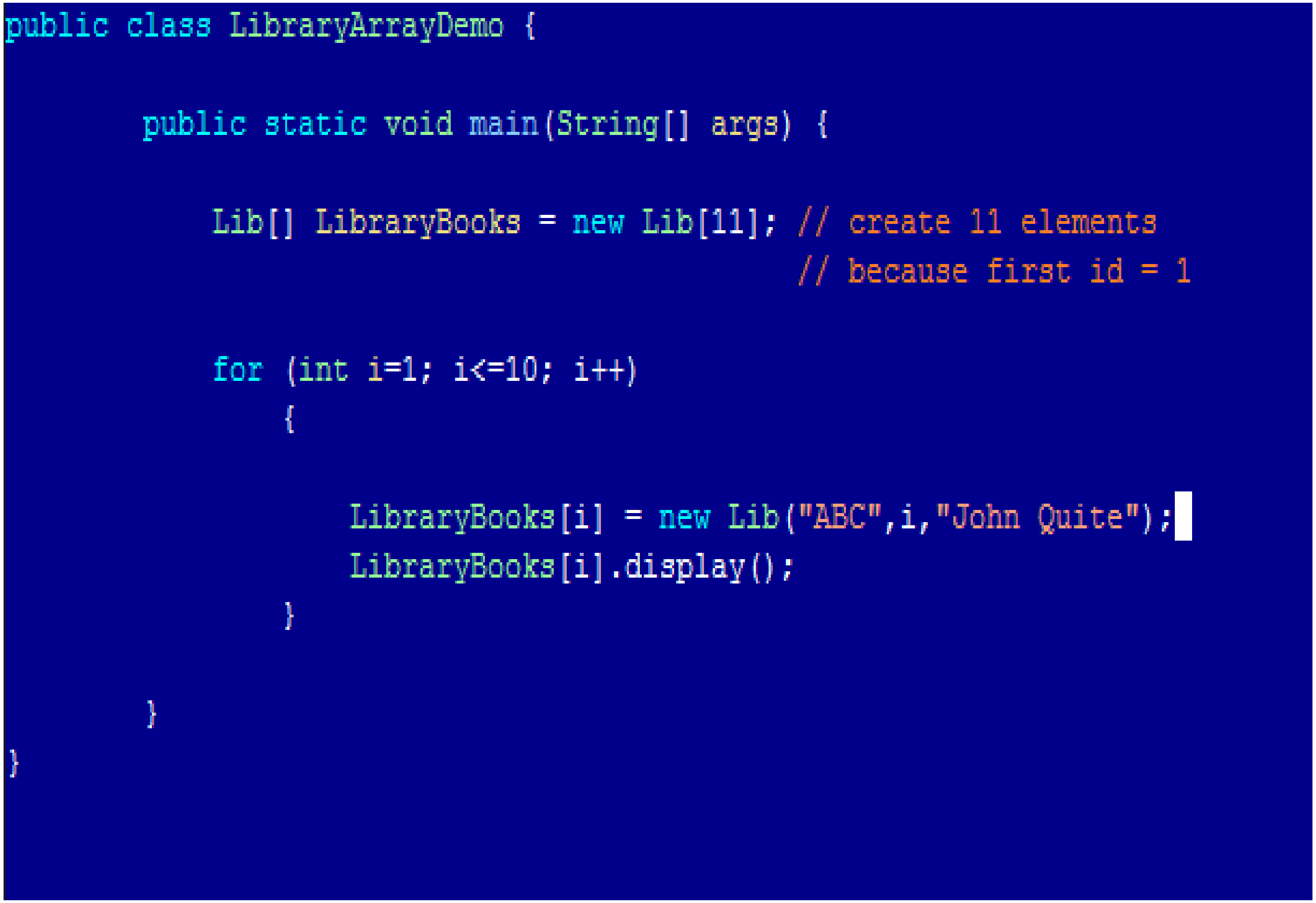
and incrementing book id……….



```java
import java.util.*;

public class Lib
{

    private String title;
    private int id;
    private String author;

    public Lib (String title, int id, String author)
    {
        this.title = title;
        this.id = id;
        this.author = author;
    }

    // accessors
    public String getTitle()
    {
        return title;
    }

    public int getId()
    {
        return id;
    }

        public String getAuthor()
    {
        return author;
    }
```

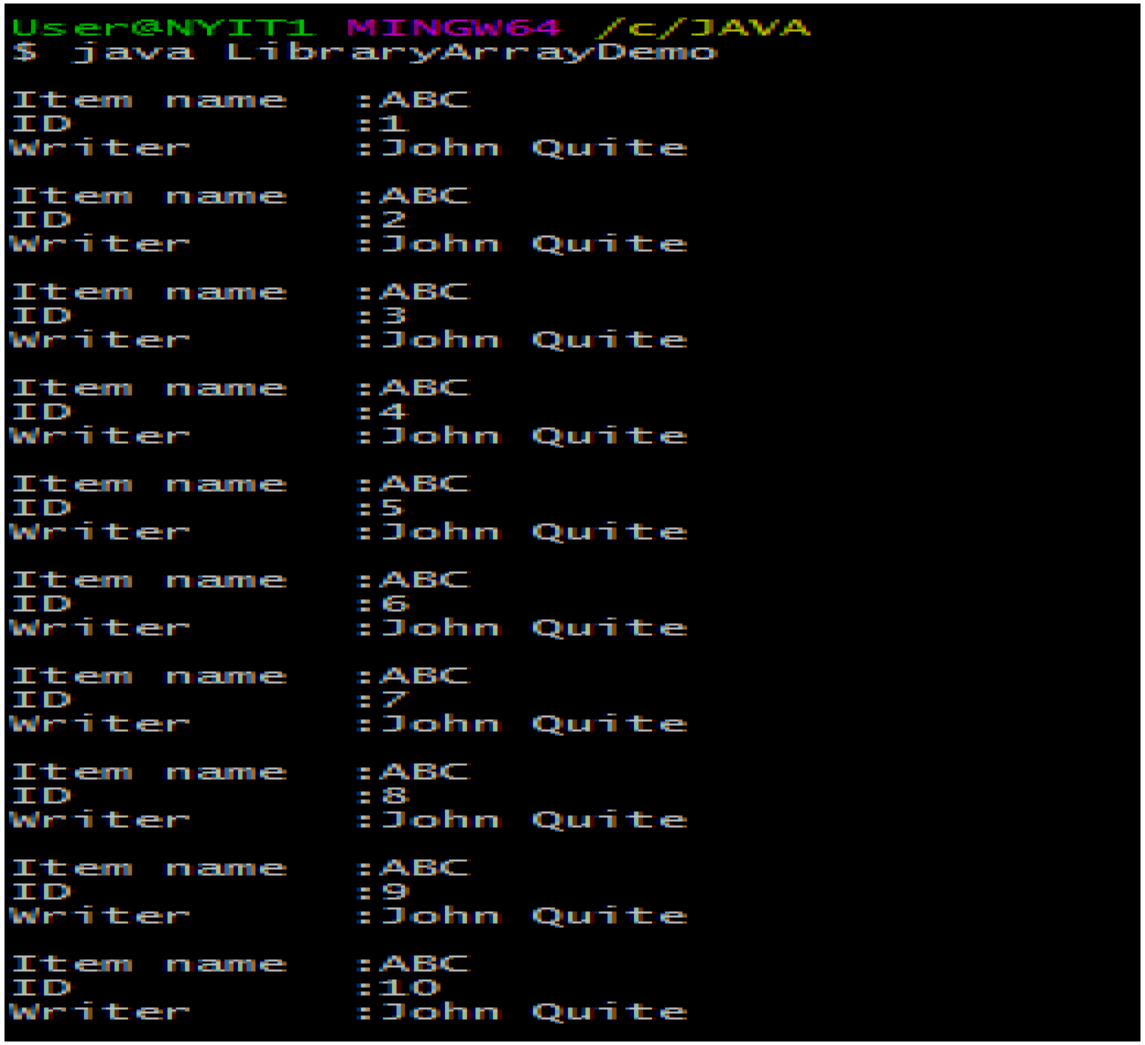


```java
    //mutators

    public void setAll (String newTitle, int newID, String newAuthor)
    {
        title  = newTitle;
        id     = newID;
        author = newAuthor;
    }


    public void display()
    {
         System.out.println();
         System.out.println("Item name  :" + title);
         System.out.println("ID         :" + id);
         System.out.println("Writer     :" + author);
    }
}
```



```
public class LibraryArrayDemo {

        public static void main(String[] args) {

            Lib[] LibraryBooks = new Lib[11]; // create 11 elements
                                              // because first id = 1

            for (int i=1; i<=10; i++)
                {

                    LibraryBooks[i] = new Lib("ABC",i,"John Quite");
                    LibraryBooks[i].display();
                }

        }
}
```

- Example screen output



```
User@NYIT1 MINGW64 /c/JAVA
$ java LibraryArrayDemo

Item name     :ABC
ID            :1
Writer        :John Quite

Item name     :ABC
ID            :2
Writer        :John Quite

Item name     :ABC
ID            :3
Writer        :John Quite

Item name     :ABC
ID            :4
Writer        :John Quite

Item name     :ABC
ID            :5
Writer        :John Quite

Item name     :ABC
ID            :6
Writer        :John Quite

Item name     :ABC
ID            :7
Writer        :John Quite

Item name     :ABC
ID            :8
Writer        :John Quite

Item name     :ABC
ID            :9
Writer        :John Quite

Item name     :ABC
ID            :10
Writer        :John Quite
```