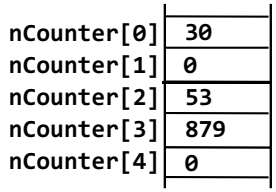
Introduction to Arrays

- Array data structures represent collections of data elements of the same data type
  - Reference variable points to a collection of variable elements grouped sequentially in memory
  - Elements accessed with same identifier (variable name) and by their element (*index*) number
  - Element numbering begins with 0 to one less then the total number of elements



 1

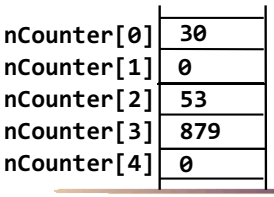
Declaring and Creating Arrays

- Array Declaration declares a variable to reference the array and specify the elements common data type
  `elementDataType[] arrayRefVariableName;`
  - Examples
    ```
    int[] nCounter;
    double[] dScore;
    String[] args;
    ```
- Array Creation is performed using the new operator which will allocate memory for the array elements
  `arrayRefVariableName = new elementType[arraySize];`
  - Examples
    ```
    nCounter = new int[12];
    dScore = new double[20];
    ```
- Array Declaration and Creation combined in one statement
  `arrayRefVariableName = new elementType[arraySize];`
  - Examples
    ```
    int[] nCounter = new int[12];
    double[] dScore = new double[20];
    ```
- Numerical array elements automatically initialized to 0 in Java

 2

Accessing Array Elements

- Elements accessed using `arrayRefVariableName` (*variable name*) and by their element (*index*) number
- Element numbering begins with 0 to one less then the total number of elements
- After array declaration and creation
  ```
  int[] nCounter = new int[12];
  ```
- Elements automatically initialized based on dataType
  - Numeric initialized to `0`
  - Character to `'\0000'`
  - Boolean to `false`
- May assign values to elements
  ```
  nCounter[0] = 30;
  nCounter[2] = 53;
  nCounter[3] = 879;
  ```



 3

for Loop used to access Array elements

- For loops can be used to access any or all elements within an array element range
- Do not attempt to access array elements out side of the element range (Example below 0 … 20)

```
1.  public class ArrayEx2
2.  {
3.    public static void main(String[] args)
4.    {
5.      int nI;
6.      char[] cGrade = new char[21];
7.      for(nI = 16; nI <= 20; nI++)
8.        cGrade[nI] = 'P';
9.      for(nI = 0; nI < 16; nI++)
10.       cGrade[nI] = 'F';
11.     for(nI = cGrade.length - 1; nI >= 0; nI--)
12.       System.out.printf("Score = %2d  Grade = %c\n", nI, cGrade[nI]);
13.   }
14. }
```

```
Score = 20  Grade = P
Score = 19  Grade = P
Score = 18  Grade = P
Score = 17  Grade = P
Score = 16  Grade = P
Score = 15  Grade = F
Score = 14  Grade = F
Score = 13  Grade = F
Score = 12  Grade = F
Score = 11  Grade = F
Score = 10  Grade = F
Score =  9  Grade = F
Score =  8  Grade = F
Score =  7  Grade = F
Score =  6  Grade = F
Score =  5  Grade = F
Score =  4  Grade = F
Score =  3  Grade = F
Score =  2  Grade = F
Score =  1  Grade = F
Score =  0  Grade = F
```

 4

Array Bounds Checking

- Java requires an array element dimension
  `char[] cGrade = new char[100];`
- Java has runtime Array Bounds Checking which prevents accessing an element that does not exist based on the dimension range
  - If attempting to access an element out of range, will throw a *Runtime Exception* (Chapter 13)
  - `ArrayIndexOutOfBoundsException`
- The array length property returns the total number of elements contained in an array.

```
for(nI = 16; nI < cGrade.length; nI++)
    cGrade[nI] = 'P';
```

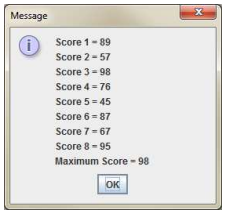

Array Initialization

- Arrays can be declared, created, and initialized within one statement
  - May continue across multiple lines
  - Must initialize all array elements, can't skip

```
1.  public class ArrayEx2
2.  {
3.    public static void main(String[] args)
4.    {
5.      int nI;
6.      char[] cGrade = {'F','F','F','F','F','F','F','F','F',
7.        'F','F','F','F','F','F','F','P','P','P','P','P'};
8.      for(nI = cGrade.length - 1; nI >= 0; nI--)
9.        System.out.printf("Score = %2d  Grade = %c\n",
10.       nI, cGrade[nI]);
11.   }
12. }
```


Counter Controlled Array Processing

```
1.  import javax.swing.JOptionPane;
2.  public class ArrayEx3
3.  {
4.    public static void main(String[] args)
5.    {
6.      int nI, nMax, nScore[] = new int[8];
7.      String sOutput = "";
8.      for(nI = 0; nI < nScore.length; nI++)
9.      {
10.       nScore[nI] = Integer.parseInt(
11.       JOptionPane.showInputDialog(null,
12.          "Enter Score " + (nI + 1) + ":"));
13.     }
14.     for(nI = 0, nMax = 0; nI < nScore.length; nI++)
15.     {
16.       sOutput += "Score "+(nI+1)+" = "+nScore[nI]+"\n";
17.       if(nScore[nI] > nMax) nMax = nScore[nI];
18.     }
19.     JOptionPane.showMessageDialog(null, sOutput
20.       + "Maximum Score = " + nMax);
21.   }
22. }
```



Java Array Class

- Documentation available in API docs and Chapter 2
  http://docs.oracle.com/javase/6/docs/api/java/util/Arrays.html

`java.util`

`Class Arrays`

The Arrays class provides many useful static methods for performing functions on declared arrays that are passed by reference as parameters.

Method Summary

`static void sort(char[] a)` Sorts the specified array of chars into ascending numerical order.
`static void sort(char[] a, int fromIndex, int toIndex)` Sorts the specified range of the specified array of chars into ascending numerical order.
`static void sort(double[] a)` Sorts the specified array of doubles into ascending numerical order.
`static void sort(int[] a)` Sorts the specified array of ints into ascending numerical order.
`static String toString(char[] a)` Returns a string representation of the contents of the specified array.
`static String toString(double[] a)` Returns a string representation of the contents of the specified array.
`static String toString(int[] a)` Returns a string representation of the contents of the specified array.
`static void fill(int[] a, int val)` Assigns the specified int value to each element of the specified array of ints.
`static void fill(int[] a, int fromIndex, int toIndex, int val)` Assigns the specified int value to each element of the specified range of the specified array of ints.
`static boolean equals(double[] a, double[] a2)` Returns true if the two specified arrays of doubles are *equal* to one another.
`static void fill(int[] a, int fromIndex, int toIndex, int val)` Assigns the specified int value to each element of the specified range of the specified array of ints.

for-each Loop Array Processing

- for each element of array traverses array sequentially

```
for( elementType element : arrayRefVar )
```

```
1.  import javax.swing.JOptionPane;
2.  public class ArrayEx3A
3.  {
4.    public static void main(String[] args)
5.    {
6.      int nScore[] = new int[3]; // Quantity of Scores
7.      String sOutput = "";
8.      for(int nJ = 0; nJ < nScore.length; nJ++)
9.        nScore[nJ] = Integer.parseInt(
10.         JOptionPane.showInputDialog(null, "Enter Score: "));
11.     for (int nElement: nScore)
12.       System.out.println(nElement);
13.     java.util.Arrays.sort(nScore);
14.     for(int nI=0; nI < nScore.length; nI++)
15.       sOutput += "Score "+(nI+1)+" = "+nScore[nI]+"\n";
16.     JOptionPane.showMessageDialog(null,
17.       "Array sorted from low to high score\n"+sOutput);
18.   }
19. }
```


Sentinal Controlled Processing

```
1.  import javax.swing.JOptionPane;
2.  public class ArrayEx4
3.  {
4.    public static void main(String[] args)
5.    {
6.      int nI=0, nCnt, nSum=0, nScore[]=new int[50];
7.      String sDisplay = "";
8.      do
9.      {
10.       nScore[nI] = Integer.parseInt(
11.         JOptionPane.showInputDialog(null,
12.           "Enter Score "+(nI+1)+": (-1 to quit)"));
13.     nI++;
14.     }while(nScore[nI-1] >= 0);
15.     java.util.Arrays.sort(nScore);
16.     nCnt = nI - 1;
17.
18.     for(nI = nScore.length - 1; nScore[nI] > 0; nI--)
19.     {
20.       sDisplay += "Score "+ (50 - nI) +" = "+nScore[nI]+"\n";
21.       nSum += nScore[nI];
22.     }
23.     JOptionPane.showMessageDialog(null, sDisplay
24.       + "\nAverage Score = " + (float)nSum/nCnt);
25.   }
26. }
```

Arrays and Method Parameters

- Arrays can be passed to methods
  - Pass-by-Value is for primitive datatype variables, Not Arrays
  - Pass-by-Sharing is Pass-by-Reference in most languages
  - Array element values can be altered in method
- Method Definitions with Array Parameters
  - `public static void initializeCards(int[] nCards){ }`
  - `public static void displayCard(int nI, int[] nCd, char[] cRk, char[] cSt){ }`
  - `public static double[] getDimensions(String... sPrompt){ }`
- Method Calls with return value passing array
  - `initializeCards(nDeck);`
  - `System.out.print("Dealer Hand: ");`
    `for(int nD = 0; nD < 5; nD++)`
    `displayCard(nD, nDeck, cRanks, cSuits);`
  - `double[] dDim = getDimensions("Diameter");`
    `dDim = getDimensions("Length", "Width", "Height" );`


```
1.  public class DeckOfCards {
2.     public static void main(String[] args) {
3.        int[] nDeck = new int[52];
4.        char[] cSuits = {'S', 'H', 'D', 'C'};
5.        char[] cRanks = {'A','2','3','4','5','6','7','8','9','T','J','Q','K'};
6.        initializeCards(nDeck);
7.        shuffleCards(nDeck);
8.        System.out.print("Dealer Hand: ");
9.        for(int nD = 0; nD < 5; nD++)
10.         displayCard(nD, nDeck, cRanks, cSuits);
11.       System.out.print("\n  Your Hand: ");
12.       for(int nD = 5; nD < 10; nD++)
13.         displayCard(nD, nDeck, cRanks, cSuits);
14.    }
15.    public static void initializeCards(int[] nCards) {
16.       for(int nI = 0; nI < nCards.length; nI++)
17.         nCards[nI] = nI;
18.    }
19.    public static void shuffleCards(int[] nCard) {
20.       for(int nI = 0; nI < nCard.length; nI++)
21.       {
22.         int nIndex = (int)(Math.random() * nCard.length);
23.         int nTemp = nCard[nI];
24.         nCard[nI] = nCard[nIndex];
25.         nCard[nIndex] = nTemp;
26.       }
27.    }
28.    public static void displayCard(int nI, int[] nCd, char[] cRk, char[] cSt) {
29.       char cSuit = cSt[nCd[nI] / 13];
30.       char cRank = cRk[nCd[nI] % 13];
31.       System.out.printf("%c%c ", cRank, cSuit);
32.    }
33. }
```

```
Dealer Hand: 5H 4S 3H 8H QC
  Your Hand: TS 2S 4H 4C JH
```

Variable Length … Method Parameters

- The Ellipsis … operator specifies a variable-length parameter list
  - Must be last parameter in a list
  - Only one variable-length parameter may be specified per method
  - Java treats variable length operator as an array similar to initialization
- Method Definition with Parameter using …

```
1.  public static double[] getDimensions(String... sPrompt)
2.   {
3.     double[] dDimensions = new double[4];
4.     String sEntry;
5.     for(int nI=0; nI < sPrompt.length; nI++)
6.     {
7.       sEntry = JOptionPane.showInputDialog("Enter the "
8.         + sPrompt[nI] +':', "0");
9.       dDimensions[nI] =  Double.parseDouble(sEntry);
10.    }
11.    return dDimensions;
12.  }
```

- Method Calls with return value passing array

```
1.  double[] dDim = getDimensions("Length", "Width", "Height" );
2.  dDim = getDimensions("Diameter", "Height");
3.  dDim = getDimensions("Diameter");
```

 13

```
1.   public static double[] getDimensions(String... sPrompt) {
2.      double[] dDimensions = new double[4];
3.      String sEntry;
4.      for(int nI=0; nI < sPrompt.length; nI++) {
5.        sEntry = JOptionPane.showInputDialog("Enter the "+ sPrompt[nI] +':', "0");
6.        dDimensions[nI] =  Double.parseDouble(sEntry);
7.      }
8.      return dDimensions;
9.    }
10.   public static char selectVolArea(String sShape) {
11.     char cChoice;
12.     while(true) {
13.        String sEntry = JOptionPane.showInputDialog("Calculate which of the "
14.          +"following:\n[v]=Volume\n[s]=Surface Area\n[b]=Both\n[c] = Cancel","");
15.        cChoice = Character.toLowerCase(sEntry.charAt(0));
16.        if(cChoice == 'v' || cChoice == 's' || cChoice == 'b' || cChoice == 'c')
17.          break;
18.        else
19.          JOptionPane.showMessageDialog(null, "'"+ cChoice
20.            + "' not listed, Please try again!");
21.     }
22.     return cChoice;
23.   }
24.   public static void selectRect() {
25.     String sUnits, sDisplay;
26.     char cVolArea;
27.     cVolArea = selectVolArea("Rectangular Solid");
28.     if(cVolArea == 'v' || cVolArea == 's' || cVolArea == 'b') {
29.       sUnits = JOptionPane.showInputDialog("Enter the units of measure", "cm");
30.       double[] dDim = getDimensions("Length", "Width", "Height");
31.       sDisplay = "The rectangular solid with dimensions:\nLength="+dDim[0]
32.         +"  Width=" + dDim[1] + "  Height=" + dDim[2];
33.       if(cVolArea == 'v' || cVolArea == 'b')
34.         sDisplay += "\nVOLUME = "+ dDim[0]*dDim[1]*dDim[2] +" cubic "+sUnits;
35.       if(cVolArea == 's' || cVolArea == 'b')
36.         sDisplay += "\nSURFACE AREA = "+ 2*(dDim[0]*dDim[1] + dDim[1]*dDim[2]
37.           + dDim[0]*dDim[2]) + " square " + sUnits;
38.       JOptionPane.showMessageDialog(null, sDisplay);
39.     }
40.   }
```

Two-Dimensional Arrays

- Arrays can have two or more dimensions
  - Two dimensional Arrays

```
int [][] nStateHighs = new int [50][12];
```

  - Visualize as table with rows and columns
- Example Monthly high temperatures for all 50 states

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
[0]												
[1]												
[2]	66	64	72	78	85	90	99	105	98	90	88	80
.												
.												
.												
[48]												
[49]												

stateHighs [2] [7] row 2, col 7 might be Arizona's high for August 105

 15

Homework 3: Sorted Scores

1. For this program you will enter a series of nine test scores with a possible range of scores between 0 and 100.
2. The program needs to load the scores into array elements and then displays the sorted scores from lowest to highest. You should use the Arrays.sort() method.
3. After displaying the nine sorted scores the program will display the high, low, mean, and median score for the entered scores.
4. You should go through the usual program design phase, but you need to submit only the test data.
5. Implement your program design using NotePad++ or Eclipse and name your file *YourName*_hw3.java
6. Compile using Java SE 6 JDK compiler and debug until all syntax errors are eliminated. Demonstrate your code runs without logic errors by running the program and enter your known test data.
7. Upload via WebTycho your *YourName*_hw3.java program source code file to the Homework 3 assignment folder. Submit on paper: Cover sheet, your known test data, output for test data, and source code.
8. This program is due at the beginning of Class 2 - Week 6.

 16