Java Fundamentals
6-1: Arrays
Practice Activities

Lesson Objectives:

- Write a single-dimensional array in a Java program using primitive data types
- Write a single-dimensional array in a Java program using reference (Object) types
- Write a 2-dimensional array in a Java program using primitive data types
- Write a 2-dimensional array in a Java program using reference (Object) types
- Declare an array, initialize an array, and traverse the array
- Describe array initialization
- Distinguish between the String method length() and an array's length value
- Rewrite a Java program to store integers into an array, perform a mathematical calculation, and display the result
- Use alternative array declaration syntax

Vocabulary:

Identify the vocabulary word for each definition below.

	The act of progressing through an array
	A structure that stores multiple values of the same data type
	A two-dimensional array.
	An integer that identifies the location of a value in an array
	The ability to pass data into the main function and access it as an element of an array.
	A logical computational procedure that if correctly applied ensures the solution of a problem
	An array of arrays, similar to a table, matrix, or spreadsheet.
	A for loop inside of a for loop
	A named object used to store more than one value.

Try It/Solve It:

1. Declare a one dimensional array name score of type int that can hold 9 values.
2. Declare a 2-dimensional array named price of type float that has 10 rows and 3 columns.
3. Declare and initialize a 2-dimensional array named **matrix** of type long that has 4 rows and 3 columns to have all it's values set to 5.
4. Declare and initialize a one dimensional byte array named **values** of size 10 so that all entries contain 1.
5. Without typing in the code determine the output of the following program.

```
int num[] = {7,7,6,6,5,5,4,4};
for(int i = 0; i < 8; i = i + 2)
        System.out.print(num[i]);
```

6. Without typing in the code determine the output of the following program.

```
int[][] num = {{3,3,3},{2,2,2}};
int[] array = {4,3,2};
for(int i = 0; i < 3; i++){
        num[1][i] = num[0][i]+array[i];
}

for(int i = 0; i < 2; i++){
        for(int j = 0; j < 3; j++){
                System.out.print(num[i][j]);
        }
        System.out.println();
}
```

7. In a certain class, there are 5 tests worth 100 points each. Write a program that will take in the 5 tests scores for the user, store the tests scores in an array, and then calculate the students average.
8. In Algebra class we learn about matrices. We learn to add, subtract, and multiply 2x2 matrices and 3x3 matrices. Below are some examples from Algebra class with the answers:

$$\begin{pmatrix}3 & 4\\ 5 & 6\end{pmatrix}+\begin{pmatrix}1 & 0\\ -2 & 3\end{pmatrix}=\begin{pmatrix}4 & 4\\ 3 & 9\end{pmatrix}$$

$$\begin{pmatrix}3 & 4\\ 5 & 6\end{pmatrix}-\begin{pmatrix}1 & 0\\ -2 & 3\end{pmatrix}=\begin{pmatrix}2 & 4\\ 7 & 3\end{pmatrix}$$

$$\begin{pmatrix}3 & 4\\ 5 & 6\end{pmatrix}*\begin{pmatrix}1 & 0\\ -2 & 3\end{pmatrix}=\begin{pmatrix}-5 & 12\\ -7 & 18\end{pmatrix}$$

It is almost apparent how to add. We add the first position in the first matrix with the first position in the second matrix. We continue with the corresponding positions to get the answer. Subtraction follows this same positional methods. Multiplication of matrices appears to be confusing since it does not follow the positional method used in addition and subtraction.

The answer is achieved by taking the row from the first matrix and the column from the second matrix and multiplying the respective values and then taking the sum of the products.

The answer above was achieve as follows:

3(1)+4(-2)=-5 3(0)+4(3)=12

5(1)+6(-2)=-7 5(0)+6(3)=18

Write a program that take in two matrices and then allow the user to choose to add, subtract, or multiply them and display the answer. The program will display the following menu:

A. Enter Matrix A

B. Enter Matrix B

C. Display A + B

D. Display A - B

E. Display A * B

F. Exit

The program should loop and allow the user to continue to choose different options until they choose quit. The well written program will modularize the process into different methods.

9. Use the following code to implement a Deck Object. When finished, add the following features:

A. Add a method shuffle to the Deck Class. Call the method from the Main class to verify that the deck is indeed shuffled.

B. Add a Hand Class that contains an array of 5 Card references. Have the program Deal the Hand two cards and display them for the user. Tell the user how many points they have and ask them if they would like another card or not. Continue to allow the player to add cards until they reach 5 cards, or the total is greater than 21.

C. Adjust the Card class to allow Aces to count as 11 to start. If the Hand Class has a value greater than 21, have the Hand Class check for Aces and reduce their point value to 1.

D. Have the program create a dealer Hand that the user can play against. The user should try to get as close to 21 without going over in an effort to beat the Dealer. If the Dealer has 16 or more the Dealer should stop taking cards.

```
public class Main {
    public static void main(String args[]){
                    Deck d = new Deck();
                    d.print();
     }

}

public class Deck {
    Card[] cardArray = new Card[52];
    Deck(){ //constructor
              int suits = 4;
              int cardType = 13;
              int cardCount = 0;
       for(int i = 1; i <= suits; i++)
              for(int j = 1; j <= cardType; j++){
                        cardArray[cardCount] = new Card(i,j);
                        cardCount++;
              }
      }
            public void print(){
                    for(int i = 0; i < cardArray.length; i++)
                              System.out.println(cardArray[i]);
            }
}

public class Card{
          String suit,name;
          int points;
    Card(int n1, int n2){
              suit = getSuit(n1);
              name = getName(n2);
              points = getPoints(name);
    }
```

```
    public String toString(){
        return "The " + name + " of " + suit;
    }
    public String getName(int i){
            if(i == 1)  return "Ace";
            if(i == 2)  return "Two";
            if(i == 3)  return "Three";
            if(i == 4)  return "Four";
            if(i == 5)  return "Five";
            if(i == 6)  return "Six";
            if(i == 7)  return "Seven";
            if(i == 8)  return "Eight";
            if(i == 9)  return "Nine";
            if(i == 10)  return "Ten";
            if(i == 11)  return "Jack";
            if(i == 12)  return "Queen";
            if(i == 13)  return "King";
            return "error";
    }
    public int getPoints(String n){
      if(n == "Jack" ||n == "Queen" ||n == "King"||n == "Ten")
            return 10;
      if(n == "Two")
            return 2;
      if(n == "Three")
            return 3;
      if(n == "Four")
            return 4;
      if(n == "Five")
            return 5;
      if(n == "Six")
            return 6;
      if(n == "Seven")
            return 7;
      if(n == "Eight")
            return 8;
      if(n == "Nine")
         return 9;
      if(n == "Ace")
         return 1;
      return -1;
     }
    public String getSuit(int i){
            if(i == 1)  return "Diamonds";
            if(i == 2)  return "Clubs";
            if(i == 3)  return "Spades";
            if(i == 4)  return "Hearts";
            return "error";
    }
}
```