Practices for Lesson 8

Practices Overview

In these practices, you use `for` loops and `while` loops to process data within arrays or ArrayLists. Two challenge practices are included here for those who have extra time and wish to be challenged.

Practice 8-1: Writing a Class that Uses a `for` Loop

Overview

In this practice, you create the `Counter` class that uses a simple `for` loop to print a sequence of numbers.

Assumptions

The `CounterTest.java` file appears in the practice folder for this lesson, `Lesson08`

Tasks

1. Create a new project from existing source called Practice09. Set the **Source Package Folder** to point to `D:\labs\les09`. Remember to also change the Source/Binary Format property. If you need further details, refer to Practice 1-2, Step 3.
2. Create a new Java class called "Counter". Declare and initialize a `public final int` field called `MAX_COUNT`. Assign the value 100 to this field.

   Hint: Use the keyword `final` to designate this is as a constant field.
3. Create a method called displayCount that does the following:
   - Counts from 1 to the value of the MAX_COUNT constant, using a `for` loop. Increment the value of the loop variable by 1.
   - Displays the value of the loop variable if it is divisible by 12. Display this on a single line, separated by a space.

   Hints
   - Example of a for loop:

     ```
     for (int i= 1; i < 10; i++) // loops 9 times
     ```
   - Use the modulus operator (%) to check divisibility by 12. If it is divisible by 12, the result of the modulus operation will be zero.
   - Use the `System.out.print` method to keep all displayed values on the same line.

   Solution:

   ```
   public void displayCount(){
        for(int count = 1; count <= MAX_COUNT; count++){
             if (count % 12 == 0) {
                  System.out.print(count + " ");
             } // end if
        } // end for
   } // end method
   ```
4. Save and compile your program. Test it by running the CounterTest class.
5. You should receive the following list of numbers as an output:

   ```
   12 24 36 48 60 72 84 96
   ```

Practice 8-2: Writing a Class that Uses a `while` Loop

Overview

In this practice, you write a class named `Sequence` that displays a sequence starting with the numbers 0 and 1. Successive numbers in the sequence are the sum of the previous two numbers. For example: 0 1 1 2 3 5 8 13 21… This sequence is also called the Fibonacci series.

Assumptions

The `SequenceTest.java` file appears in the practice folder for this lesson, `Lesson08`, and consequently in your project.

Tasks

1. Create a new Java class called "Sequence" with three fields called `firstNumber`, `secondNumber`, and `nextNumber`. Assign the values of 0 and 1 to the `firstNumber` and `secondNumber` fields, respectively. Also declare a `public final int` called SEQUENCE_LIMIT. Set its value to 100.
2. Create a method called `displaySequence`. Use the following high-level steps to code the method. If you need more help, detailed instructions are provided following these steps:
   a. Print the value of `firstNumber` and `secondNumber` to start the sequence. Separate all numbers in the sequence by a space.
   b. Calculate the sum of `firstNumber` and `secondNumber` and assign the sum to `nextNumber`.
   c. Create a `while` loop with the following characteristics:
      - *boolean expression*: Repeat if the value of `nextNumber` is less than or equal to SEQUENCE_LIMIT.
      - *code block*:
        - Print the value of `nextNumber`.
        - Assign the value of `secondNumber` to `firstNumber` and the value of `nextNumber` to `secondNumber`.
        - Recalculate the value of `nextNumber` to be the sum of `firstNumber` and `secondNumber`.
   d. After the while loop, use the System.out.println method to create a new line.

   Detailed instructions:

   a. Before the `while` loop begins, use the `System.out.print` method to print `firstNumber` and `secondNumber`, concatenating a space to the end of each variable in your `print` statements.
   b. Set `nextNumber` equal to `firstNumber` + `secondNumber`.
   c. Start a `while` loop that evaluates the following expression in determining whether to loop again:

      ```
      while(nextNumber <= SEQUENCE_LIMIT)
      ```

      - Within the while block, do the following:
        - Print the `nextNumber` field. Add a space to the end of it.
        - Set `firstNumber` equal to `secondNumber`, and `secondNumber` equal to `nextNumber`.

- Set `nextNumber` equal to `firstNumber` + `secondNumber`.

d. Outside the while loop block, use `System.out.println` to create a new line for the "Build Successful…" message that appears after the sequence.

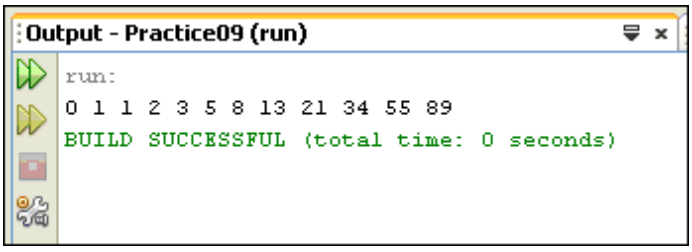
Solution:

```
public class Sequence{
     public int firstNumber = 0;
     public int secondNumber = 1;
     public int nextNumber;
     public final int SEQUENCE_LIMIT = 100;

     public void displaySequence(){
          // Print the first two numbers
          System.out.print(firstNumber + " ");
          System.out.print(secondNumber + " ");
          // Calculate the next number
          nextNumber = firstNumber + secondNumber;

          while(nextNumber <= SEQUENCE_LIMIT){
               // Print the next number of the sequence
               System.out.print(nextNumber + " ");
               firstNumber = secondNumber; // new first number
               secondNumber = nextNumber; // new second number
               // Calculate the next potential number
               nextNumber = firstNumber + secondNumber;
          } // end of while
          // Finish it off with a carriage return
          System.out.println();
     } // end of method
} // end of class
```

3. Save and compile your program. Run the SequenceTest class to test it.



Challenge Practice 8-3: Converting a `while` Loop to a `for` Loop

This practice is optional. Check with your instructor for recommendations about which optional practices to do. Perform this practice only if you are certain that you have enough time to perform all of the non-optional practices.

Overview

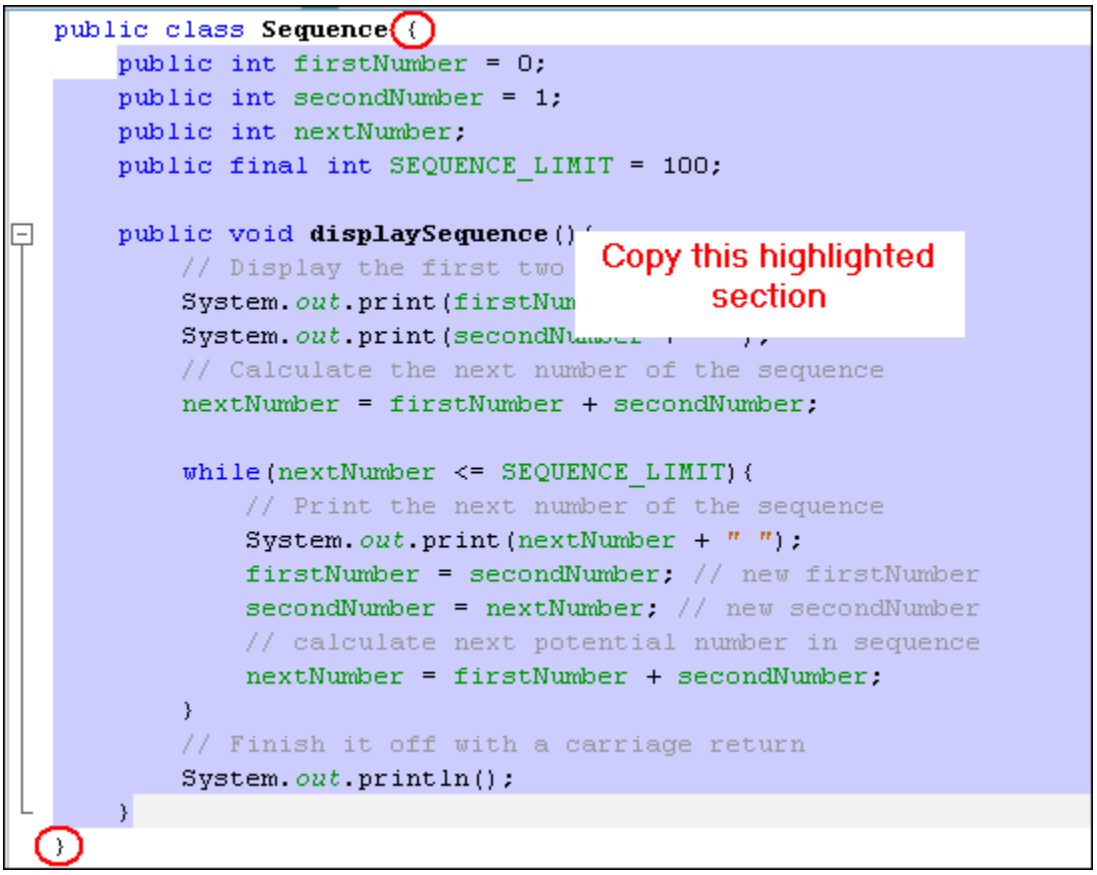
In this practice, you create a new class, ChallengeSequence, based on the Sequence class you created in the last practice. You modify the `displaySequence` method to use a `for` loop instead of a `while` loop.

Assumptions

This practice assumes that you have completed Practice 8-2. It also assumes that the ChallengeSequenceTest.java file appears in the practice folder for this lesson, `Lesson08`, and consequently in your project.

Tasks

1. Create a new Java class called "ChallengeSequence". Copy all the code that occurs between the outer (class) brackets of the Sequence class and paste it inside the outer brackets of the ChallengeSequence class.




```
public class Sequence {
    public int firstNumber = 0;
    public int secondNumber = 1;
    public int nextNumber;
    public final int SEQUENCE_LIMIT = 100;

    public void displaySequence(){
        // Display the first two
        System.out.print(firstNu
        System.out.print(secondNu
        // Calculate the next number of the sequence
        nextNumber = firstNumber + secondNumber;

        while(nextNumber <= SEQUENCE_LIMIT){
            // Print the next number of the sequence
            System.out.print(nextNumber + " ");
            firstNumber = secondNumber; // new firstNumber
            secondNumber = nextNumber; // new secondNumber
            // calculate next potential number in sequence
            nextNumber = firstNumber + secondNumber;
        }
        // Finish it off with a carriage return
        System.out.println();
    }
}
```

2. Create an additional final field called `SEQUENCE_COUNT` and assign a value of 10 to it. Be sure that you don't change any of the other field names.
3. In the `displaySequence` method, modify the `while` loop to a `for` loop such that only the first 10 values ot the Fibonacci series are displayed.

Hints

- Remember that the first two numbers in the sequence are displayed before the loop begins. Your `for` loop must display the remaining eight values.
- There are a several ways of handling the discrepancy between the `SEQUENCE_COUNT` value and the number of values that need to be displayed within the loop. One approach is to adjust the initial count in the loop.

One Possible Solution:

```
public void displaySequence(){
     System.out.print(firstNumber + " ");
     System.out.print(secondNumber + " ");
     nextNumber = firstNumber + secondNumber;

     for(int count = 2; count < SEQUENCE_COUNT; count++){
     // Start at 2 and loop until you get to 9 (8 numbers)
          System.out.print(nextNumber + " ");
          firstNumber = secondNumber;
          secondNumber = nextNumber;
          nextNumber = firstNumber + secondNumber;
     }
     System.out.println();
}
```

4. Save and compile your program. Run the ChallengeSequenceTest class to test your code. Your output should display the following series:

```
0 1 1 2 3 5 8 13 21 34
```

Practice 8-4: Using `for` Loops to Process an ArrayList

Overview

In this practice, you create two new methods in two different classes. One of the methods uses a traditional `for` loop to display the values in an `ArrayList`. The other method uses an *enhanced* `for` loop to display the values in the `ArrayList`. This practice contains two sections:

- Using a `for` Loop with the VacationScaleTwo Class
- Using an Enhanced `for` loop with the NamesListTwo Class

Assumptions

This practice assumes that the following files appear in the practice folder for this lesson, `Lesson08` and consequently, in your project:

- VacationScaleTwo.java
- VacationScaleTwoTest.java
- NamesListTwo.java

Using a `for` Loop with the VacationScaleTwo Class

1. Open the VacationScaleTwo class in the editor. This is similar to the VacationScale class you wrote for the practices in Lesson 8, except an `ArrayList` is used to store vacation days instead of an array.
2. Add a new method called `displayVacationDays`. High-level instructions for this task are provided in the table below. More detailed instructions can be found following the table.

Step	Code Description	Choices or Values
a.	Use a `for` loop to loop through the elements of the `vacationDays` ArrayList.	Use the `size` method of the `ArrayList` in the boolean expression that determines the end of the loop.
b.	Within the loop, display each value of the `ArrayList` and its position in the list with a suitable label.	Use the `get` method of the `ArrayList` together with `System.out.println` to display the value.

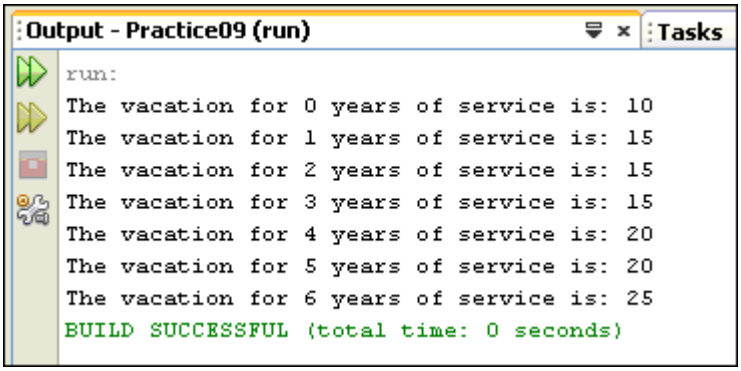
   a. In the `displayVacationDays` method, add a `for` loop with the following criteria:

   ```
   for(int years = 0; years < vacationDays.size(); years++)
   ```

   b. In the `for` loop block, use `System.out.println` to print the value of each `ArrayList` element. Use the `get` method of the `ArrayList`, passing the `years` variable as an argument. It references the current index number of the `vacationDays` list.

   ```
   System.out.println("The vacation for " + years +
    " years of service is: " + vacationDays.get(years));
   ```

3. Save and compile your program, and then run the VacationScaleTwoTest class to test it. You should see an output similar to this:



Using an Enhanced `for` Loop with the NamesListTwo Class

4. Open the NamesListTwo class in the editor. This is similar to the NamesList class that you saw in Lesson 8. It has only one method, `setList`. This method initializes the `ArrayList` and then prints the size of the list.
5. Add a new method to the NamesListTwo class called `displayNames`. You use an *enhanced* `for` loop in this method to process the `ArrayList`. High-level instructions for this task are provided in the table below. More detailed instructions can be found following the table.

Step	Code Description	Choices or Values
a.	Display an introductory message to describe the list that follows.	
b.	Start the enhanced `for` loop (remember that an `ArrayList` is defined to hold elements of type `Object`)	`for (Object name : listOfNames)`
c.	In the `for` block, display the current element of the `ArrayList`.	`Use the name reference`

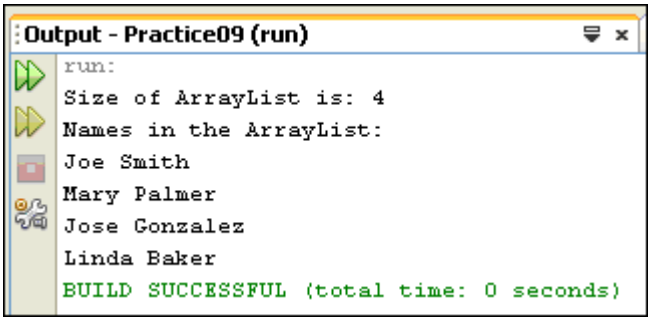
   a. Use the `System.out.println` method to print the message:
      `"Names in the ArrayList: "`
   b. Start the enhanced `for` loop as follows:
      `for (Object name : listOfNames)`

      Note: The `name` variable is a reference to the current element in the `listOfNames` ArrayList for each iteration of the `for` loop.
   c. Within the `for` loop block use `System.out.println` to print the `name` reference.

Solution:

```
public void displayNames(){
     System.out.println("Names in the ArrayList: ");
     for(Object name : listOfNames){
          System.out.println(name);
     }
}
```

6. Create a new Java *Main* Class called NamesListTwoTest.
7. In the `main` method, do the following:
   a. Declare and initialize a local variable of type `NamesListTwo` called `names`.
      `NamesListTwo names = new NamesListTwo();`
   b. Invoke the `setList` method of the `names` object.
   c. Invoke the `displayNames` method of the `names` object.
8. Save and compile your program. Run the NamesListTwoTest class to test it.
9. You should see an output from the program similar to the screenshot below:



Practice 8-5: Writing a Class that Uses a Nested `for` Loop to Process a Two Dimensional Array

Overview

In this practice, you create and process a two-dimensional array using a nested `for` loop (one loop within another loop). This practice is based on the scenario of a classroom. A classroom has 12 desks arranged in a rectangular grid comprising three rows and four columns. Students are allocated a desk at the position found vacant first, by traversing each row.

The following table shows the class map as a grid. Each cell represents a desk. Each cell contains the coordinates of the desk position in the class map.

XXXX	Column 1	Column 2	Column 3	Column 4
Row 1	0,0	0,1	0,2	0,3
Row 2	1,0	1,1	1,2	1,3
Row 3	2,0	2,1	2,2	2,3

Assumptions

This practice assumes that the `ClassMapTest.java` file appears in the practice folder for this lesson, `Lesson08`, and consequently in your project.

Tasks

1. Create a new Java class called "ClassMap" .
2. In the class, declare two public fields as follows:

```
public String[][] deskArray;
public String name;
```

3. Create a new method called `setClassMap`. In this method, initialize the `deskArray` to have three rows and four columns:

```
deskArray = new String[3][4];
```

4. Create another new method called `setDesk`. This method assigns a new student (identified by the `name` field that is set by the ClassMapTest) to an empty desk in the class map. Define the method according to the steps below:
   a. Traverse the `deskArray` to identify the first vacant element in it. Use a nested `for` loop for this purpose. For example:

```
for(int row=0; row<3; row++){
     for(int col=0; col<4; col++){
          if(deskArray[row][col]==null){
```

   b. If you find a `null` value in the `deskMap` (in other words, if you find an empty desk), assign the value of the `name` field to the vacant element.
   c. Within the inner `for` loop (the one that iterates over the columns of a row), set a local `boolean` variable (`flag`) to `true` if you assigned the name to an element. Check the value of the `flag` variable in the last line of the outer `for` loop. If it is `true`, there is no need to continue looping through the rest of the rows, so `break` out of the `for` loop.

Similarly, you can check the value of the `flag` after the close of the outer `for` loop. If the value is still false, it means that all desks are taken (no null values were found). Print a message indicating that all desks are occupied.

d. Print the position of the desk for the student and exit out of the loops. Use a `break` statement to branch out of a running loop.

Solution:

```
public void setDesk() {
     boolean flag= false;
     for(int row=0; row<3; row++){ // start of row loop
        for(int col=0; col<4; col++){ // start of column loop
          if(deskArray[row][col]==null){
             deskArray[row][col] = name;
             System.out.println
                (name +" desk is at position: Row:"
                  + row + " Column:"+col);
             flag = true;
             break; // drop out of column loop
          } // end of if
        } // end of inner/column for loop
        if (flag == true){
          break; // drop out of row loop
        } // end of if
     } // end of row for loop
     if (flag == false){
        System.out.println("All desks occupied.");
     } // end of if
} // end of method
```

Note: You test this code a little later.

5. Create another new method called `displayDeskMap`. In this method, traverse the `deskArray` in the same way you did in the last step. For each element in the array, print the name in that element (or print "null"). The output should be in the form of grid.

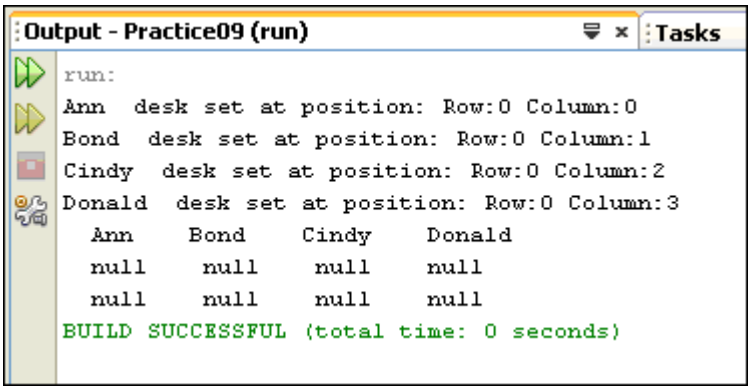
   Hint: Use a combination of `print` and `println` method calls to achieve the grid format. The grid should look similar to this:

   Ann Bond Cindy Donald
   null null null null
   null null null null

Solution:

```
public void displayDeskMap() {
     for(int row = 0; row < 3; row++){
        for(int col = 0; col < 4; col++){
            System.out.print(" "+ deskArray[row][col] +" ");
        }
        System.out.println(); // carriage return between rows
     }
}
```

6. Save and compile your program.
7. Open the ClassMapTest class and examine the code in the `main` method. It first calls `setClassMap` to initialize the array. Next it assigns a value to the `name` field of the `myClassMap` object and then invokes `setDesk`. It does this four times, with a different `name` value each time. Finally, it invokes `displayDeskMap`.
8. Run the ClassMapTest class to test your program.



9. If you do not plan to perform Practice 9-6 (an optional challenge practice), close the Practice09 project in NetBeans now.

Challenge Practice 8-6: Adding a Search Method to the ClassMap Program

This practice is optional. Check with your instructor for recommendations about which optional practices to perform.

Overview

In this practice, you add another method to the ClassMap class. This method searches through the `deskArray` to find a certain name.

Assumptions

This practice assumes that you have completed Practice 8-5.

Tasks

1. In the ClassMap class, add another new method called `searchDesk`.
2. In the `searchDesk` method, do the following:
   a. Create a nested `for` loop to traverse through the `deskArray`.
   b. If the array element is not `null`, compare the value of the `name` field with the value of the element. For example:

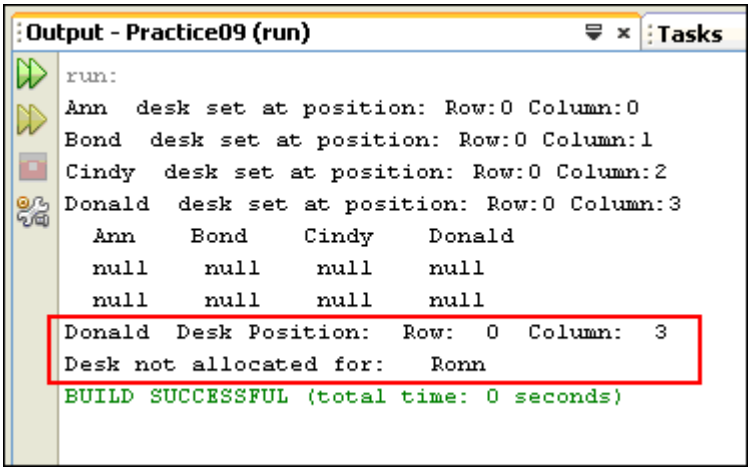
```
if(deskArray[row][col] != null &&
     deskArray[row][col].equals(name)){
```

   c. Print the position of the desk if the names are equal.
   d. Print an error message if the name is not found in the `deskArray`.
   e. Use the `break` statement to branch or exit out of the loops wherever required.

Solution:

```
public void searchDesk() {
     boolean flag= false;
     for(int row=0; row<3; row++){
        for(int col=0; col<4; col++){
           if(deskArray[row][col] != null &&
              deskArray[row][col].equals(name)){
                 System.out.println
                 (name +" Desk Position: Row: "+row+" Column: "
                    +col);
              flag = true;
              break;
           } // end of if
        } // end of column loop
        if (flag == true){
           break;
        } // end of if
     } // end of row loop
     if (flag == false){
        System.out.println("Desk not allocated for: "+name);
     } // end of if
} // end of method
```

3. In the ClassMapTest class, uncomment the lines of code that set the `name` value of `myClassMap` object and invoke its `searchDesk` method (this combination occurs twice).
4. Save and compile your program. Run the ClassMapTest class to test the program.



5. This is the conclusion of the Lesson 8 practices. Close the Practice09 project now.