proprep

Workbook

Table of Contents

Further Programming Concepts

Arrays

Questions

1) Create a new project, and within it a class called arraysApp. Inside the class create a method called outputOddNumbers()

a. Inside the method, declare an array of bytes with the following elements:
   i. 1, 2, 4, 5, 10, 11, 12, 14, 17, 20, 21

b. Using a for loop, iterate through the array outputting to the screen each element of the array on a new line

c. Modify the code inside the loop so that numbers are output only if they are odd

2) Create another method in the same class, called topThree()

a. Inside the method, declare an array of bytes called scores with the following elements:
   i. 55, 75, 12, 67, 23, 95, 23, 61, 82, 43, 37

b. Declare a second, empty array of bytes, called results with a size of 3

c. Use a loop to iterate through the scores array, comparing each element value with what is stored in results[0]. If the current value during iteration is higher than what is already stored in results[0], replace results[0] with the current value

d. results[0] should now contain the largest value in the array. After the loop, output to the screen the message "The highest score is:" followed by the contents of results[0]

e. Add a second loop after the above code which finds the second highest value by comparing the current element with results[1], AND results[0]. If it is higher than results[1] but lower than results[0], replace results[1] with the current value.

f. results[1] should now contain the second largest value in the array. Output to the screen the message "The 2nd highest score is:" followed by the contents of results[1]

g. Add a final loop after all previous code, which finds the third highest value and stores it into results[2]

h. results[2] should now contain the third largest value in the array. Output to the screen the message "The third highest score is:" followed by the contents of results[2]

i. Change the first element in the scores array from 55 to 97 and check your program still works, with new numbers appearing as the top 3 scores

3) Create a new method called `common()`

a. Declare two arrays, `arrayA` and `arrayB`, containing the following values
   i. `arrayA`: 4, 7, 10, 16, 25, 28, 40, 53, 79, 92, 111
   ii. `arrayB`: 2, 5, 6, 10, 12, 19, 28, 33, 44, 51, 87, 92

b. Use a loop to iterate through `arrayA`, checking whether each element is present in `arrayB`. When an element is found in both arrays,
output the value alongside the message "common value found".

4) Create a method called `lookup()`

a. Declare two arrays:
   i. an integer array called `gradeBoundaries`, with initial values: 35,43,50,65,77
   ii. a char array called `grades`, with initial values: 'U', 'E', 'D', 'C', 'B', 'A'

b. Declare three variables:
   i. an integer called `mark`, with an initial value of 22
   ii. a boolean called `found`, with an initial value of false
   iii. an integer called `position`, with an initial value of 0

c. Create a do…while loop which ends when `position` is equal to the length of the `gradeBoundaries` array or when `found` is equal to true

d. Inside the do…while loop, see if the value of `mark` is less than the value of the element in the `gradeBoundaries` array currently being pointed to by the index variable `position`
   i. If it is, assign the value `true` to the `found` variable
   ii. If it is not, increment the `position` variable

e. Outside the loop print the element in the `grades` array that is pointed to by the index variable `position`

f. Test the code outputs the character 'U'

g. Change the value of `marks` to 43, and 77 to confirm grades of D and A are output respectively.

5) Below is pseudo code for an algorithm called a Bubble sort:

```
numbers= [9, 5, 4, 15, 3, 8, 11]
numItems = len (numbers)
for i = 0 to numItems -2
   for j = 0 to (numItems - i - 2)
      if (numbers[j] > numbers[j+1]) then
         temp = numbers [j]
         numbers[j] = numbers[j + 1]
         numbers[j + 1] = temp
      end if
   next j
   output (numbers)
   output ("\n")
next i
```

a. Turn this pseudo code into a Java method called bubbleSort
b. The line `output(numbers)` will need to print the entire array on one line using a loop, and the line `output ("\n")` will output a new line character
c. Change the numbers array by adding the number 2 as the last element after the number 11, and test the bubbleSort method again. How many times does the number 2 get swapped?

6) Create a method called twoD()

a. Declare a two dimensional integer array called `grid`, with 5 rows and 7 columns
b. Using a pair of nested loops, assign to each element of the array a value according to this formula: `element value = column index + (row index * 7)`
c. After the nested loops in part b which write values to the array, create another pair of nested loops.
   i. In the inner loop output each element from the array on the same line using `System.out.print`
   ii. After each element is output, output the tab character `"\t"` again on the same line using `System.out.print`
   iii. After the inner loop, but inside the outer loop, output a new line character
d. Test your program outputs the numbers 0 to 34
e. Make a change to the program so it outputs the numbers 1 to 35 instead
f. Change the program again, so that when the initial size of the array is changed, from 5 rows, 7 columns to 8 rows, 5 columns, the numbers 1 to 40 are printed.

7) Create a method called `transpose`()

a. Declare a 2D char array initialised with the following values:
{'h', 'i', 's', 't', 's'}, {'e', 's', 'e', ' ', 'a'},
{'r', ' ', 'c', 'm', 'g'}, {'e', 'a', 'r', 'e', 'e'}, {' ', ' ', 'e', 's', '.'}

b. Declare a pair of integer variables, `row` and `column`, both initialised to `0`

c. Create a pair of nested loops, with the inner loop iterating `column`, and the outer loop iterating `row`

d. Inside the inner loop, output the element pointed to by `row` and `column`, using System.out.print so all characters appear on the same line

e. Test your program outputs the characters as seen above in part a in the order that they appear in the array by row

f. Swap the order of the loops created in c) so that column is iterated in the outer loop, and row is iterated in the inner loop. Test again, can you explain the output you see?

Answer Key

To view the answers to these exercises please refer to the appropriate videos.

Strings and String Processing

Questions

1) Create a new project and within it a class called `stringApp`

a. Create a new method called `reverseString()`

b. Declare a string variable and initialise it with the literal string `"stressed"`

c. Use a loop to iterate through each character starting from the back of the string, outputting each character as it is encountered, all on one line using `System.out.print`

d. Test the program by running it, if it works correctly it will print the input string reversed. Try some other input strings like `"spam"` and `"peels"`, all three should output familiar words.

2) Create a new method called `robot()`

a. Declare:

   i. a string variable called `commands` and initialise it with the literal string `"FFLFFRFFLFFFFRRFFH"`

   ii. a char variable called `currentCommand` with no initial value

b. Using a loop to iterate through each character in the string, extract the current character into `currentCommand`

c. Inside the loop, display a message as follows depending on the contents of `currentCommand`:

   i. F – Forward

   ii. R – Right turn 90 degrees

   iii. L – Left turn 90 degrees

   iv. H – Halt

d. Test your code produces the expected output

3) Create a new method called `extractInitials`()

a. Declare a string variable `fullName` containing the initial value `"Robert Womack"`
b. Declare an integer variable `position`, with `0` as the initial value
c. Declare a string variable `initials` with an empty string "" as the initial value
d. Add the first character of `fullName` to the `initials` variable using the concatenation operator
e. Use a loop to iterate through each character in the string
   i. each time a space or a hyphen is found, extract the next character and add it to `initials`
   ii. Keep going until no more characters are left in the `fullName` string
f. Output the contents of the `fullName` and `initials` strings separated by a suitable character or message
g. Test the program with the following initial values for `fullName`
   i. "Robert Womack"
   ii. "Penny Martha Lane"
   iii. "David Michael-Cox"
   iv. "George Frederick Ernest Albert Saxe-Coburg-Gotha"

4) Create a new method called `parseAssembly`()

a. Declare:
   i. a string variable with the initial value `"LDA 101 STA 100 NOP LDA 102 NOP HLT"`
   ii. a string variable called `strMatch` with the initial value "LDA"
b. Split the input string on spaces, into a new array so that each substring is stored in a new array element. For example "LDA" will be stored in array element 0, `101` in array element 1 etc.
c. Using a loop, inspect each element in the array to see if it contains the same value as `strMatch`. If it does output the following all on the same line:
   i. the contents of `strMatch`
   ii. a space character
   iii. the contents of the next element in the array
   iv. a new line character
d. Test your program outputs `LDA 101` and `LDA 102` and nothing else
e. Change the initial value of `strMatch` to `"STA"` and check the output is `STA 100`

5) Create a method called `isGraphicFileType()`

a. Declare:
   1. a String array called `files` containing the following values: `"MyThesis.doc", "profilePic.jpg", "budget.xls", "favjpg.pdf", "funny.gif", "temp", "logo.png", "photo.jpeg"`
   2. an String variable called `filetype`
   3. an integer variable called `position`

b. Iterate through the elements of the array outputting the value of each element without adding a newline character at the end

c. Inside the loop below the output line in part b,
   1. use a suitable string method to find whether the string contains a "." and to store the position where it is found into `position`
   2. If the "." was found, extract a substring into `filetype` so `filetype` contains the characters after the "."
   3. Compare the substring with the file extensions "jpg", "jpeg", "gif" and "png". If there is a match, output `"\t * is a graphic file type *"` without a newline
   4. Output a newline character inside the loop, but outside the selection structure in parts ii-iii

d. Test the method outputs all the filenames, with an indication that the file is a graphic file type for only the relevant files.

e. Change the file type for "logo.png" to "logo.pdf" and confirm the difference in output.

6) Create a method called `isPalindrome`()

a. Declare a string variable called `inString` with the initial value "Did Hannah see bees? Hannah did."

b. Create two more string variables called `rawString` and `reverseRawString`, both with initial values of ""

c. Copy the contents of `inString` to `rawString` using regular expressions and Java string functions so that:
   1. any upper case letters in `rawString` are forced to lower case
   2. only letters are in `rawString` and all other characters are ignored

d. Using a loop, output the `rawString` characters into `reverseRawString` in reverse order, i.e. the first character in `reverseRawString` will be the last character in `rawString`

e. Compare `rawString` to `reverseRawString`
   1. if they are equal output `inString` followed by the message " is a palindrome"
   2. if they are not equal output `inString` followed by the message " is not a palindrome"

f. Test your program works, it should find that the input string is a palindrome. Remove one letter from the initial value to see if it now finds it is not a palindrome

7) Create a new method called `findSongs`()

a. Declare:
   i. a string variable called `sourceString` with an initial value as follows: "<album>Greatest<song trackid=1>Song for You</song><song trackid=2>Sad Song</song><song>Another Song</song></album>"
   ii. a string array called `songs` with 10 uninitialised elements
   iii. integer variables called `index`, `start` and `end`, initialised to 0
   iv. a boolean variable called `finished`, initialised to `false`

b. Using a loop which terminates when `finished` is `true`, parse `sourceString` to extract the song name between `<song>` and `</song>`. Note some songs have extra text between `<song` and `>`
   i. Use `start` and `end` to store the beginning and end of the substring to be extracted
   ii. each substring extracted should be stored into the next element in the `songs` array using `index`
   iii. when there are no more songs to be found, set `finished to true`

c. Outside the parse loop, use another loop to output the contents of the `songs` array, with each element appearing on a new line in the output.

d. Test your method stores three strings into the `songs` array

e. Using the format `"<song>`*`name of song`*`</song>"` add two more songs at the end of the initial value of `sourceString` and test whether these additional songs appear in the `songs` array and therefore the output

Answer Key

To view the answers to these exercises please refer to the appropriate videos.