**Assignment 4 Turn in Sheet Name:**

Lab Questions (**Total 50 Pts. + 4 EXTRA Pts.**)

Big Java, Late Objects / Java for Everyone, 2e

**Chapter Number: 6 Arrays and ArrayLists**

1) (5 pts.) Write a method that is passed an array, x, of doubles and an integer rotation amount, n. The method creates a new array with the items of x moved forward by n positions. Elements that are rotated off the array will appear at the end. For example, suppose x contains the following items in sequence:

1 2 3 4 5 6 7

After rotating by 3, the elements in the new array will appear in this sequence:

4 5 6 7 1 2 3

Array x should be left unchanged by this method. Use the following code to help you get started. Be sure to test your program with different rotation amounts.

public class Arrays

{

public static void main(String[] args)

{

double[] x = {8, 4, 5, 21, 7, 9, 18, 2, 100};

System.out.println("Before rotation: ==============================");

for (int i = 0; i < x.length; i++)

{

System.out.println("x[" + i + "]: " + x[i]);

}

x = rotate(x, 3);

System.out.println("After rotation: ==============================");

for (int i = 0; i < x.length; i++)

{

System.out.println("x[" + i + "]: " + x[i]);

}

}

public static double[] rotate(double[] x, int k){

reverse(x, 0, x.length -1);

reverse(x, 0, k-1);

reverse(x, k, x.length-1);

return x;

}

public static void reverse(double[] x, int start, int end){

while(start < end){

swap(x, start, end);

start++;

end--;

}

}

public static void swap(double[] a, int pos1, int pos2){

double temp = a[pos1];

a[pos1]= a[pos2];

a[pos2] = temp;

}

}

2.1) (5 pts.) Create a class called CustomerLister with a main method that instantiates an array of String objects called customerName. The array should have room for five String objects. Use an initializer list to put the following names into the array:

Cathy

Ben

Jorge

Wanda

Freddie

Print the array of names.

|  |
| --- |
| public class CustomerLister {    public static void main(String[] args)  {  String[] customerName = {"Cathy", "Ben", "Jorge", "Wanda", "Freddie"};    for(String display : customerName)  System.out.println(display);  }    }  **OUTPUT:**  **Cathy**  **Ben**  **Jorge**  **Wanda**  **Freddie** |

2.2) (5 pts.)Use new to create a second double array called customerBalance in the main method. Allow room for five customer balances, each stored as a double. In the loop that prints each customer name, add some code to prompt the user to enter a balance for that customer. Read the keyboard input with a Scanner object. Use the following balances for the input:

100.00

234.56

2.49

32.32

400.00

After all the balances have been entered, print out each customer and his/her balance.

|  |
| --- |
| public class CustomerLister {    public static void main(String[] args)  {  Scanner input = new Scanner(System.in);  String[] customerName = {"Cathy", "Ben", "Jorge", "Wanda", "Freddie"};  double[] customerBalance = new double[5];    for(int i = 0; i < customerName.length; i++)  {  System.out.print("Input balance for " + customerName[i] + ": ");  customerBalance[i] = input.nextDouble();  }    for(int i = 0; i < customerName.length; i++)  System.out.println(customerName[i] + "\t:\t$" + customerBalance[i]);  }    }  **OUTPUT:**  **Input balance for Cathy: 100**  **Input balance for Ben: 234.56**  **Input balance for Jorge: 2.49**  **Input balance for Wanda: 32.32**  **Input balance for Freddie: 400.00**  **Cathy : $100.0**  **Ben : $234.56**  **Jorge : $2.49**  **Wanda : $32.32**  **Freddie : $400.0** |

3) (20 pts.) British puzzle maker H. E. Dudeney concocted an interesting puzzle about a bored postman called the “Peevish Postman Problem”. According to Dudeney, the postman worked in a small post office with consecutive letter boxes numbered 1 to 100. Each box was equipped with a door that could be opened and closed. Late one evening the postman made a “pass” through the boxes and opened every door. Still bored, he walked back to the beginning and made a second pass, this time visiting boxes 2, 4, 6, …, 100. Since those doors were now open, he closed them. On the third pass he visited boxes 3, 6, 9, 12, …, 99 and if a door was open he closed it, and if the door was closed he opened it. He continued to make passes through the boxes and always followed the same rule: On each pass *i* from 1 to 100, he visited only boxes that were multiples of *i*, … and changed the state of each door he visited. After making 100 passes at the doors, he surveyed the results and was surprised by the pattern of doors that he saw.

Use a Boolean array to represent the doors. A true value in the array represents an open door, and a false value represents a closed one. You will have to write two nested loops in order to manipulate the array as described above. The inner loop will control the door number visited on a single pass, and the outer loop will control the number of passes. Print the state of each door after the 100th pass.

*Hint:* The door numbers (1-100) do not match the index of the array, so you need to consider this in your algorithm

|  |
| --- |
| public class Peevish\_Postman\_Problem {    public static void main(String[] args)  {  boolean[] doors = new boolean[101]; //defaulted to false;  for(int i = 1; i < 101; i++)  for(int j = 1; j < 101; j++)  if(j % i == 0)doors[j] = !doors[j];      System.out.println("Doors that are open: ");  for(int j = 1; j < 101; j++)  if(doors[j] == true) System.out.print(j + " ");        }  }  **OUTPUT:**  **Doors that are open:**  **1 4 9 16 25 36 49 64 81 100** |

The puzzle was conceived as a paper and pencil entertainment. Can you explain the pattern of doors?

**The output of the problem, or the doors that are ‘open’ at the end of the 100 passes are all the perfect squares in sequential order. IE 1­2, 22, 32, 52, 62, 72, 82, 92, 102**

4) (9 pts.) Array lists are objects that, like arrays, provide you the ability to store items sequentially and recall items by index. Working with array lists involves invoking ArrayList methods, so we will need to develop some basic skills. Let’s start with the code below:

import java.util.ArrayList;

public class ArrayListRunner

{

public static void main(String[] args)

{

ArrayList<String> names = new ArrayList<String>();

System.out.println(names);

}

}

The main method imports java.util.ArrayList and creates an ArrayList that can hold strings. It also prints out the ArrayList and, when it does, we see that the list is empty: [ ].

Complete the following tasks by adding code to this skeleton program. If you are asked to print a value, provide a suitable label to identify it when it is printed.

a) Invoke add() to enter the following names in sequence: Alice, Bob, Connie, David, Edward, Fran, Gomez, Harry. Print the ArrayList again.

b) Use get() to retrieve and print the first and last names.

c) Print the size() of the ArrayList.

d) Use size() to help you print the last name in the list.

e) Use set() to change “Alice” to “Alice B. Toklas”. Print the ArrayList to verify the change.

f) Use the alternate form of add() to insert “Doug” after “David”. Print the ArrayList again.

g) Use an enhanced for loop to print each name in the ArrayList.

h) Create a second ArrayList called names2 that is built by calling the ArrayList constructor that accepts another ArrayList as an argument. Pass names to the constructor to build names2. Then print the ArrayList.

i) Call names.remove(0) to remove the first element. Print names and names2. Verify that Alice B. Toklas was removed from names, but not from names2.

|  |
| --- |
| public class ArrayListRunner  {  public static void main(String[] args)  {  ArrayList<String> names = new ArrayList<String>();  names.add("Alice");  names.add("Bob");  names.add("Connie");  names.add("David");  names.add("Edward");  names.add("Fran");  names.add("Gomez");  names.add("Harry");  System.out.println("Part A: " + names);  System.out.println("Part B: " + names.get(0) + ", " + names.get(7));  System.out.println("Part C: " + names.size());  System.out.println("Part D: " + names.get(names.size()-1));  names.set(0, "Alice B. Toklas");  System.out.println("Part E: " + names);  names.add(4, "Doug");  System.out.println("Part F: " + names);  System.out.print("Part G: ");  for(String output : names)  System.out.print(output + ", ");  ArrayList<String> names2 = new ArrayList<String>(names);  System.out.println("\nPart H: " + names2);  names.remove(0);  System.out.println("Part I: " + names);  System.out.println("Part I: " + names2);  }  }  **OUTPUT:**  **Part A: [Alice, Bob, Connie, David, Edward, Fran, Gomez, Harry]**  **Part B: Alice, Harry**  **Part C: 8**  **Part D: Harry**  **Part E: [Alice B. Toklas, Bob, Connie, David, Edward, Fran, Gomez, Harry]**  **Part F: [Alice B. Toklas, Bob, Connie, David, Doug, Edward, Fran, Gomez, Harry]**  **Part G: Alice B. Toklas, Bob, Connie, David, Doug, Edward, Fran, Gomez, Harry,**  **Part H: [Alice B. Toklas, Bob, Connie, David, Doug, Edward, Fran, Gomez, Harry]**  **Part I: [Bob, Connie, David, Doug, Edward, Fran, Gomez, Harry]**  **Part I: [Alice B. Toklas, Bob, Connie, David, Doug, Edward, Fran, Gomez, Harry]** |

5) (10 pts.) Define a Class SurfBoard with a private(!) Class variable lastSerialNumber starting with 10000 and a private(!) instance variable serialNumber. The serialNumber will be set when an Object is instantiated from the class. Make the Class variable lastSerialNumber only be accessible (altered or read) from an Object (if there is no Object you won’t be able to print the lastSerialNumber for example, how do you do this?) Create the instance methods as needed. **Override the super class’ toString() function** with a function that prints out the same information like the super class function and additionally prints out: lastSerialNumber and serialNumber with their annotations.

Write a main program that stores newly created SurfBoard objects in an ArrayList and asks the user if he wants to produce a surfboard and reacts on (Y,y,N,n). When the answer is y(es) a surfboard is created and stored in the ArrayList and the user is asked again. Once the user answers n(o), the complete list of surfboards is printed out each in a single line (**requirement: enhanced for loop**) using the toString() function.

|  |
| --- |
| public class SurfBoardList{  static class surfBoard  {  private final int lastSerialNumber = 10000;  private int serialNumber;    public surfBoard(int serial)  {  serialNumber = serial;  }    @Override public String toString()  {  return "Serial Number: " + serialNumber +  "\t out of " + lastSerialNumber;  }  }  public static void main(String[] args)  {  ArrayList<surfBoard> surfBoardList = new ArrayList<surfBoard>();  Scanner uInput = new Scanner(System.in);  char answer;  int i = 1;  do{  System.out.println("Enter Y to create a surfboard or N to exit");  answer = uInput.next().charAt(0);  if(answer == 'y' || answer == 'Y')  {  surfBoard newBoard = new surfBoard(i++);  surfBoardList.add(newBoard);  System.out.println("Surfboard Created");  }  else if(answer == 'n' || answer == 'N')  System.out.println("N entered, current list of boards:");  else{  System.out.println("Error: Invalid input");  answer = 'y';}    }while (answer == 'y' || answer == 'Y');    for(surfBoard output : surfBoardList)  System.out.println(output);  }  }  **OUTPUT:**  **Enter Y to create a surfboard or N to exit**  **y**  **Surfboard Created**  **Enter Y to create a surfboard or N to exit**  **y**  **Surfboard Created**  **Enter Y to create a surfboard or N to exit**  **y**  **Surfboard Created**  **Enter Y to create a surfboard or N to exit**  **y**  **Surfboard Created**  **Enter Y to create a surfboard or N to exit**  **n**  **N entered, current list of boards:**  **Serial Number: 1 out of 10000**  **Serial Number: 2 out of 10000**  **Serial Number: 3 out of 10000**  **Serial Number: 4 out of 10000** |