Tech College

Java Week 1 Assignments

1. **Triplets**

Triplets are a set of three similar things.
Complete the function to print all the triplets <A, B, C> such that A+B = C
public static void printTriplets(int[ ] data) {
}

UTC	Sample Input	Sample Output
1	data = {2,3,4,5,7}	<2,3,5> <2,5,7> <3,4,7>
2	data = {1,2,3,4,5,7,9}	<1,2,3> <1,3,4> <1,4,5> <2,3,5> <2,5,7> <3,4,7> <4,5,9>

2. **Matrix Arrangement**

Write a function to return an array where the midst position contains the smallest value followed by the next smallest value on the right of the midst position, followed by the next smallest value to the left of the midst position and the rest of numbers continue in this format
Function signature:
public static int [ ] arrangeElements(int[ ] [] inputArray) {
// write the code here
}

UTC	Sample Input	Sample Output

1	4 2 13
	3 8 5
	9 6 17
		{17,9,6,4,2,3,5,8,13}

2	4 1 3
	3 8 5
	9 16 17
		{17,8,4,3,1,3,4,5,16}

3. Regional Transport Offices of Bangalore and Vehicle Registrations

Description:
Vehicles registration’s first two parts “XX-XX” contains the state information and the RTO zonal area code were the vehicle was registered.
For example if vehicle registration number is “KA-50-EF-1234”, then “KA” refers to “Karnataka” and “50” refers to Zonal office area “Yelahanka” and similarly if the vehicle registration is “KA-05-45”, then “KA” refers to “Karnataka” and “05” refers to Zonal office area “Jayanagar”.
Given below is the complete set of RTO Code and corresponding area names of RTO’s in Bangalore:

RTO Code	Area
KA-01	Koramangala
KA-02	Rajajinagar
KA-03	Indiranagar
KA-04	Yeshwanthpur
KA-05	Jayanagar
Ka-50	Yelahanka

RTO Code	Area
KA-51	Electronics City
KA-52	Nelamangala
KA-53	K.R.Puram
KA-54	Nagamangala
KA-55	Mysore East
KA-56	Basavakalyan
KA-57	Shantinagar

Write a method which accepts a collection of registration numbers and returns a collection of registration numbers which are sorted on RTO-Zonal area where the vehicles were registered. Note: If vehicles are from same area then they should be sorted based on registration sequence number [Part of number without RTO Code]

UTC	Input	Expected Output	Description
UTC_02_01	KA-55-AB-4555, KA-01-EF-4444, KA-04-AB-9000, KA-56-200, KA-50-T-3111, KA-02-AG-9243	KA-56-200, KA-01-EF-4444, KA-55-AB-4555, KA-02-AG-9243, KA-50-T-3111, KA-04-AB-9000	Vehicles are sorted based on area namely: **Basavakalyan, Koramangala, Mysore East, Rajajinagar, Yelahanka, Yeshwanthpur**
UTC_02_02	KA-57-DE-111, KA-51-A-9, KA-04-500, KA-02-L-41	KA-51-A-9, KA-02-L-41, KA-57-DE-111, KA-04-500	Vehicles are sorted based on area namely: **Electronics City, Rajajinagar, Shantinagar, Yeshwantpur**
UTC_02_03	KA-57-DE-111, KA-51-A-9, KA-04-500, KA-02-L-41, KA-57-AB-9011, KA-04-A-100	KA-51-A-9, KA-02-L-41, KA-57-AB-9011, KA-57-DE-111, KA-04-500, KA-04-A-100	Since Both the vehicles “KA-57-AB-9011” and “KA-57-DE-111” are from same area they are sorted based on sequence. Similarly for “KA-04-500” and “KA-04-A-100”

4. BEST SELLING FRUITS

“BestFruits” is a popular fruits shop. As part of their business, they want to maintain a report of best selling fruits for their business improvements. So,every time a consumer purchases a fruit, the best selling fruits report should be updated with the count. The report also should maintain the fruit selection trend in the descending order of the sales, ie., the most selling fruit should be at the top of the report and the least selling fruit at the bottom.

OBJECTIVES

By the end of this exercise, you should be able to:

- Use appropriate collection to store the data
- Use of Comparable interface for comparing objects
- Use algorithms provided by Java Collection framework

Write an application to accept fruit selected by the customer and update its count and re-arrange the order based on the count. If the fruit is not present in the collection add it to the end of the collection with its count as 1.

Example:

If new fruit “Pear” is added fruit details sorted in descending order of count as shown below should be displayed:

Mango	121
Apple	60
Orange	35
Pear	1

5. **Postfix expression evaluation 10 marks**

Postfix notation is a method of writing arithmetic and algebraic expressions in which all operators follow their operands (contrast this with infix notation). For example:

Infix expression	Postfix expression	Value
4 + 6	4 6 +	10
12 / 3 + 4	12 3 / 4 +	8
5 * (28 - 14) / 2 + (49 / 7)	5 28 14 - * 2 / 49 7 / +	42

The primary advantage to using postfix notation is there is no need to explicitly describe precedence using parentheses. This simplifies machine evaluation significantly.

Below is the algorithm to evaluate a postfix expression using Stack data structure.

Assume that the operands and operators are available as a sequence of parsed tokens. Read the tokens from left to right. When you see an operand, push it on the stack. When you see an operator, pop the two top operands off the top of the stack, perform the operation on those two operands, and push the result back on the stack.

For example, in the expression 7 3 − 5 8 + ∗, when we encounter the −, we compute 7 − 3, then replace the sub-expression 7 3 − by 4.

The whole expression now looks like 4 5 8 + ∗. We continue processing from left to right. This time, + is the first operator we encounter. We apply + to the two most recently seen operands, namely 5 and 8, which gives 4 13 ∗.

Finally, this expression evaluates to 13 ∗ 4, which is 52.
The state of the stack at each step is
7
7 3
4
4 5

4 5 8
4 13
52

Input
A postfix string where the operands and operators are available as a sequence of parsed tokens
Output
Output is the value of the postfix expression.

Unit Test Case ID	Sample Input	Sample Output
UTC04_01	7 3 – 5 8 + *	52
UTC04_02	12 3 / 4 +	8
UTC04_03	5 28 14 - * 2 / 49 7 / +	42

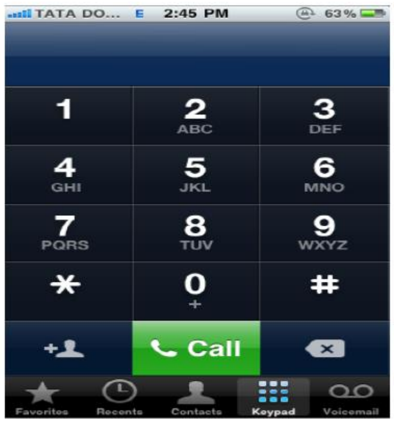
6. **String Compression**

Let us design a simple compression algorithm where only the frequency of individual letters is used to compress the data. For e.g., the string Aabcccccaaa would become a2b1c5a3. The compression logic should be applied only when the total length of the compressed string is less than the original string. For the purpose of compression logic, the case sensitiveness is not considered. For e.g. A and a are considered the same.
Input
A String that needs to be compressed is given. Given string always contains characters. The string may contain characters in upper as well as lower case.
Output
Output is the compressed string or the same string if the compressed string length is more than or equal to the length of the original string. Output contains string in lowercase always.

Unit Test Case ID	Sample Input	Sample Output
UTC03_01	aAbcccccaaA	a2b1c5a3
UTC03_02	BBBBbbb	b7
UTC03_03	ZANCKKKKPpppp	z1a1n1c1k4p5
UTC03_04	abCcc	abccc
UTC03_05	A	a
UTC03_06	ZzaA	zzaa

7. **Alpha Phrase to Phone Number Converter**

For ease of potential customers' memory, some businesses use alpha phrase for their phone numbers. For e.g.; 325-CARS can be used instead of 325-2277. Write a program that converts the given alpha phrase to the corresponding phone number. Use the picture of the key pad (given below) for your reference. Letters A, B and C represents digit 2 and so on.



Input
Input will be an alpha phrase which represents a phone number. The input may contain the spaces and the hyphen (-).
Output
Output is the phone number and it retains any space and hyphen symbol from the input.

Unit Test Case ID	Sample Input	Sample Output
UTC02_01	1800COMPUTE	18002667883
UTC02_02	9HELLO1234	9435561234
UTC02_03	MIND9999IT	6463999948
UTC02_04	1 800 GOFONTS	1 800 4636687
UTC02_05	1-800-MY-APPLE	1-800-69-27753
UTC02_06	9-HELLO-1234	9-43556-1234

REFLECTION API

The method accepts any entity object as an argument and returns an XML data as listed below:

#	Input	Output
1	Employee employee = new Employee(230,"James", "jammy@gmail.com");	<Employee> <employeeid>230</employeeid> <email>jammy@gmail.com</email> <name>James</name> </Employee>
2	Message message = new Message("Kim", "Peter", new Date(), "Will meet you during Coffee break");	<Message> <sentdate>Apr 01 2013</sentdate> <msgtext>Will meet you during Coffee break </msgtext> <sendername>Kim</sendername> <receivername>Peter</receivername> </Message>

```
        /**
         * @param args
         */
        public static void main(String[] args) {
                Employee employee = new Employee(230, "James", "jammy@gmail.com");
                String employeeXML = generateXML(employee);
                System.out.println(employeeXML);

                Message message = new Message("Kim", "Peter", new Date(), "Will meet
you during Coffee break");
                String messageXML = generateXML(message);
                System.out.println(messageXML);
        }

        private static String generateXML(Object object) {
                StringBuilder builder = new StringBuilder();
                        // complete the code
                return builder.toString();
        }
```