Java Lab Assignments

Version 1.0 | 27-FEB-2027

Table of Contents:

Table of Contents

Problem Statement 1:

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

}
```

Write JUnit test cases to test the code.

#	Sample Input	Sample Output
UTC_01_01	$\begin{array}{\|l\|l\|l\|} \hline 4 & 2 & 13 \\ \hline 3 & 8 & 5 \\ \hline 9 & 6 & 17 \\ \hline \end{array}$	[17,9,6,4,2,3,5,8,13]
UTC_01_02	$\begin{array}{\|l\|l\|l\|} \hline 4 & 1 & 3 \\ \hline 3 & 8 & 5 \\ \hline 9 & 16 & 17 \\ \hline \end{array}$	[17,8,4,3,1,3,4,5,16]

Capabilities tested:

1) Java language constructs
2) How to use and traverse through array data container
3) Unit testing and writing Unit test cases

Problem Statement **2**

Sum of consecutive elements in a subsequence

Complete the method to find the minimal length of the subsequence of consecutive elements of the sequence, sum of which is greater or equal to the specified number.

Example:

5	1	3	5	10	7	4	9	2	8

If the specified number is 15 in the sequence
5 + 1 + 3 + 5 + 10 > 15 (length is 5)
1+3+5+10 >= 15 (length is 4)
3+5+10 >= 15 (length is 3)
5 + 10 >= 15 (length is 2)
10 + 7 >=15(length is 2)
There is no element in the array whose value is >= 15 and hence the minimal length is 2

```
public static int findMinimalLength(int[] array, int value){
    //code here
```

}

#	Sample Input	Sample Output

UTC_02_01	array = {5,1,3,5,10,7,4,9,2,8}
	value = 15
		2

UTC_02_02	array = {1,2,3,4,5}
	value = 11
		3

Capabilities tested:

1) Java language constructs
2) How to use and traverse through array data container
3) Unit testing and writing Unit test cases

Problem Statement 3:

Guessing Game

Write a Java program to play the word guessing game.

The computer will pick a question and you must guess the answer. If you guess the word with less than MAXGUESS_COUNT mistakes, then you win.

The questions and answers should be stored in static array of String.

Use java.util.Scanner class to read input from keyboard.

The following suggestions might be helpful:

1. Program should choose a random question from a list of available questions.

   Sample code to generate Random Number:

```
public static int randInt(int min, int max) {
      java.util.Random rand = new java.util.Random();
      // nextInt is normally exclusive of the top value,
      // so add 1 to make it inclusive
      int randomNum = rand.nextInt((max - min) + 1) + min;
      return randomNum;
}
```

2. Program should hint how many characters are there in the answer by displaying “*” for every character. For example if the answer to the question is “ice cream”, it should display “*** *****”.
3. Each time the user makes a guess, ask the user if he/she would like to play another game.

4. The user is provided five chances to guess the correct answer; if he/she fails the program will be terminated.

Sample input/output:

```
Question: Man’s best friend
The word has 3 letter(s).
The word is: * * *

What is your guess?  You have 5 guesses remaining.
Enter: cat
Sorry, the word is not "cat".

The word is: * * *
What is your guess?  You have 4 guesses remaining.
Enter: bat
Sorry, the word is not "bat".

The word is: * * *
What is your guess?  You have 3 guesses remaining.
Enter: dog
Yes, the word is "dog"!
You made only 2 mistakes.  Good!
```

Capabilities tested:

1) Java language constructs
2) String manipulation
3) Using Scanner to read from keyboard.
4) How to use and traverse through array data container
5) Unit testing and writing Unit test cases

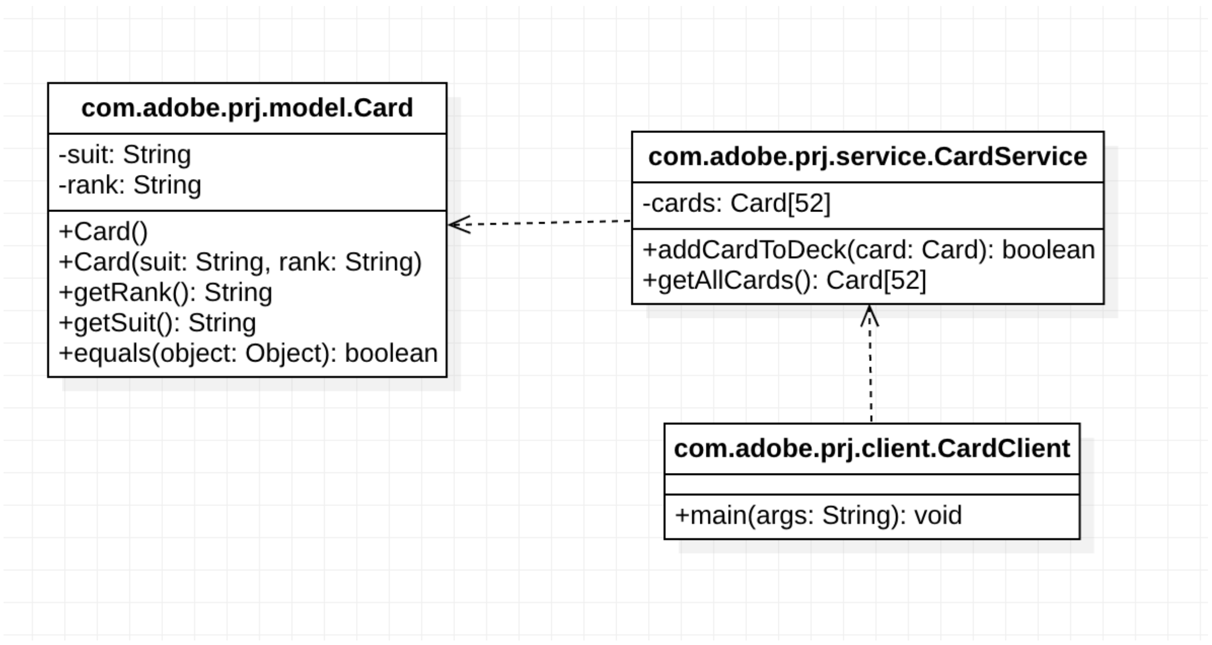
Problem Statement 4:

Playing Cards

There are 52 cards in a deck, each of which belongs to one of four suits and one of 13 ranks. The suits are **Spades**, **Hearts**, **Diamonds** and **Clubs**. The ranks are **Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen and King**. Depending on what game you are playing, the rank of the Ace may be higher than King or lower than 2.

Given:
An array of Strings to initialize suits and ranks as:

```
String [ ] suits = {“Clubs", "Diamonds", "Hearts", "Spades”};
String [ ] ranks = { "Ace", "2", "3", "4", "5", "6",  "7", "8", "9", "10", "Jack", "Queen", "King" };
```


com.adobe.prj.model.Card
-suit: String
-rank: String
+Card()
+Card(suit: String, rank: String)
+getRank(): String
+getSuit(): String
+equals(object: Object): boolean
com.adobe.prj.service.CardService
-cards: Card[52]
+addCardToDeck(card: Card): boolean
+getAllCards(): Card[52]
com.adobe.prj.client.CardClient
+main(args: String): void

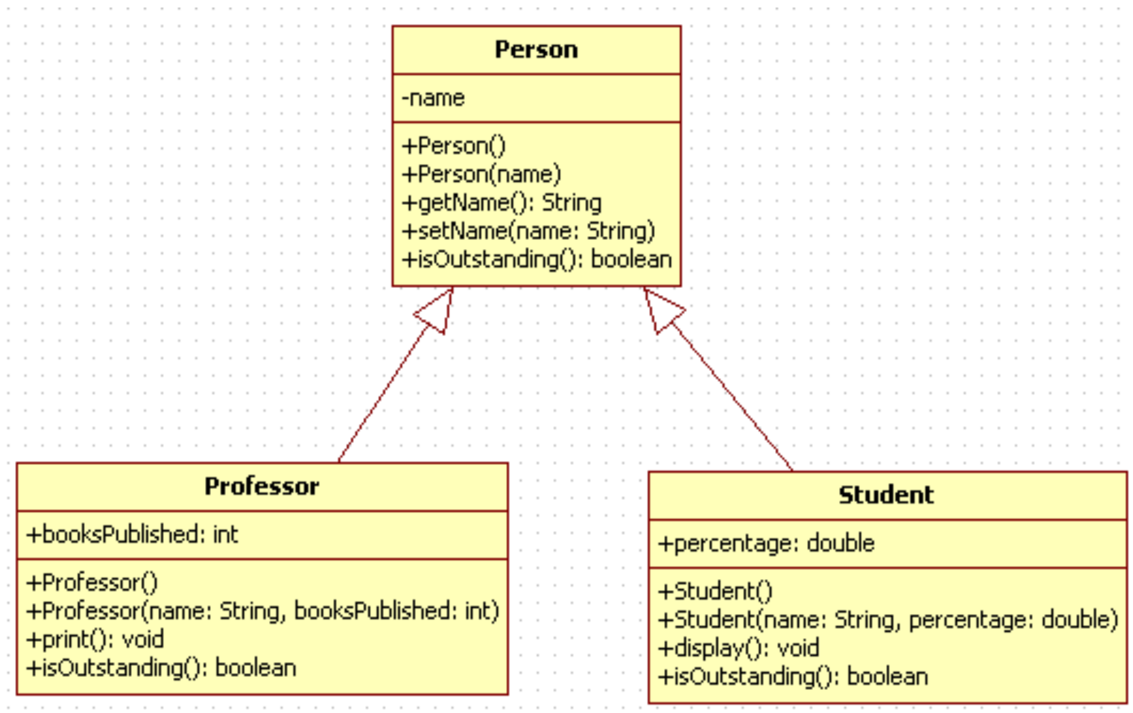
Capabilities tested:

1) Understanding OOP
2) Understanding packages
3) Uses A Relationship
4) Object identity
5) How to use and traverse through array data container
6) Unit testing and writing Unit test cases

Problem Statement 5:

Outstanding persons

Outstanding: Distinguished from others in excellence.
We judge a person if he is outstanding or not by the achievement in his/her profession



Write an application to perform the following operations:

1) Complete the entity classes shown in Class diagram
2) Implement isOutstanding( ) based on the following rules:
   a. Professor is outstanding if he has published more than 4 books
   b. Student is outstanding if his percentage is greater than or equal to 85.
3) The print ( ) method of Professor displays the name and books published by professor.
4) The display( ) method of Student displays the name and percentage of student
5) Complete the main ( ) method of PersonExample.java to implement the following:
   a. Store 5 persons in a single collection of array type (Person[ ]) , this can be a combination of students and professors
   b. Using Run-Time Type Identification print Professor and Student details.
   c. Identify all outstanding persons
6) Write Junit tests to test your code

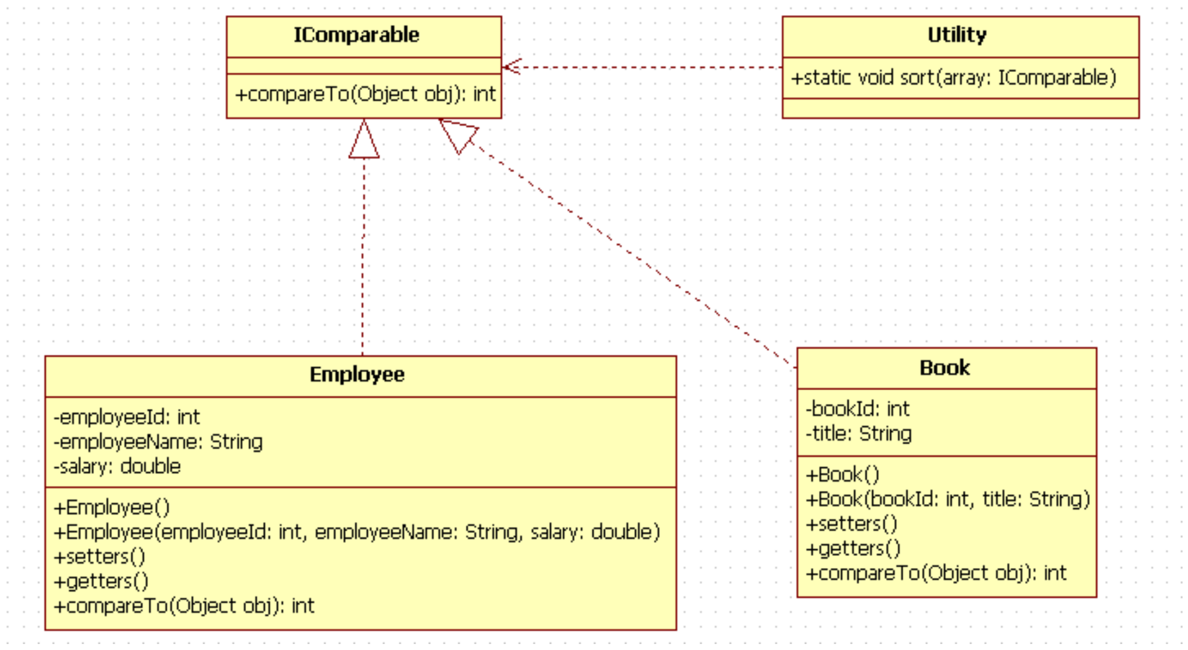
Capabilities tested:

1) Understanding Generalization and Specialization relationship
2) Overriding and dynamic binding
3) Runtime polymorphism
4) Unit testing and writing Unit test cases

Problem Statement 6:

SORTING DIFFERENT ENTITES

Sorting functionality using Open/Closed Principle: The functionality is programmed to accept an interface and is closed for modifications. If the functionality has to be used on any entity at a minimum it should implement that interface.


IComparable
+compareTo(Object obj): int
Utility
+static void sort(array: IComparable)
Employee
-employeeId: int
-employeeName: String
-salary: double
+Employee()
+Employee(employeeId: int, employeeName: String, salary: double)
+setters()
+getters()
+compareTo(Object obj): int
Book
-bookId: int
-title: String
+Book()
+Book(bookId: int, title: String)
+setters()
+getters()
+compareTo(Object obj): int

Capabilities tested:

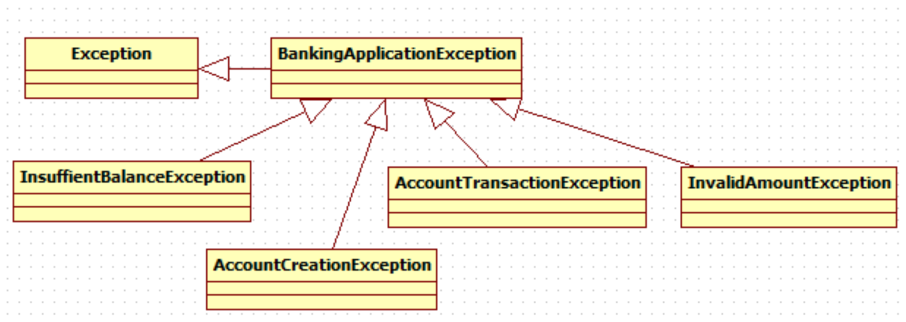
1) Understanding Realization relationship
2) Runtime polymorphism

Problem Statement 7:

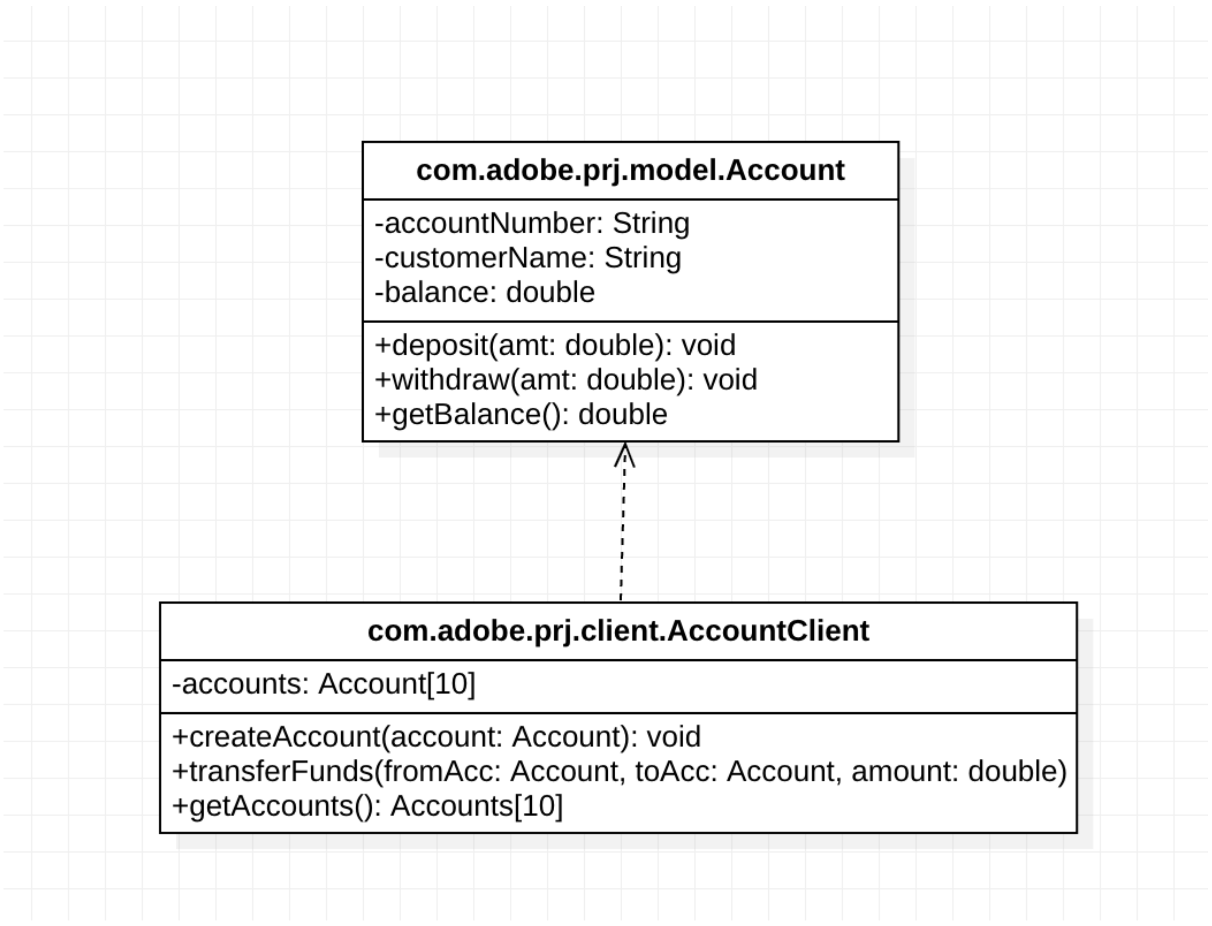
Exception handling in banking application

A simple banking application should allow account creation, transfer funds between accounts. While doing these operations many abnormalities may occur and to make your application a robust one is a challenging task.

The following exception classes are provided in "com.adobe.exception" package:


Exception
BankingApplicationException
InsuffientBalanceException
AccountTransactionException
InvalidAmountException
AccountCreationException

Implement classes as listed below to through appropriate exceptions


com.adobe.prj.model.Account
-accountNumber: String
-customerName: String
-balance: double
+deposit(amt: double): void
+withdraw(amt: double): void
+getBalance(): double
com.adobe.prj.client.AccountClient
-accounts: Account[10]
+createAccount(account: Account): void
+transferFunds(fromAcc: Account, toAcc: Account, amount: double)
+getAccounts(): Accounts[10]

Refer the class diagram and perform the following operations

1. The deposit() method should update the balance. It should throw InvalidAmountException is user tries to deposit amount <0.
2. The withdraw() method should update the balance. It should throw InvalidAmountException is user tries to deposit amount <0 and throw InsufficientBalanceException if user tries to withdraw amount more than available balance.
3. The createAccount() method of BankService is used to create a new Account. It should throw AccountCreateException if Account already exists
4. The transferFunds() method of BankService transfers funds from "fromAccount" to "toAccount". It should throw AccountTransactionException if transaction fails. Transaction fails if "fromAccount" has insuffient balance and if "fromAccount" and "toAccount" refer to the same account.

Capabilities tested:

1) Understanding OOP
2) Exception handling mechanism and writing custom exceptions

Problem Statement 8:

Product collection

any online shopping portal, the products will be displayed in order. The user interface allows the products to display sorted by brand or by price; similarly in employee portal employees can sorted by designation, last name or first name, etc.

Create a Product entity class to hold the following data:

Product ID	Brand Name	Description	Price
200	Dell	15 inch Monitor	3400.44
120	Dell	Laptop	45000.00
150	Microsoft	Windows 7	7000.50
100	Logitech	Optical Mouse	540.00

Write a program to perform the following operations

5. Stores the data shown above in a List implementation.
6. An operation to display all the products in a sorted order by default based on their product id.
7. Option for sorting based on Brand name or price based on input provided at runtime.
    a. If two products contain same brand name, description should be considered.
    b. Similarly if products have same price, product id's should be considered while displaying them in order

Sample Output if Sorted by Brand Name:

Product ID	Brand Name	Description	Price
200	Dell	15 inch Monitor	3400.44
120	Dell	Laptop	45000.00
100	Logitech	Optical Mouse	540.00
150	Microsoft	Windows 7	7000.50

Capabilities tested:

1) Understanding how to use data containers provide by java collection framework
2) Differentiate between how to implement comparison logic in the object versus the logic in client code
3) Understand provided algorithm classes

Problem Statement 9:

Best Selling Fruits

"Best Fruits" is a well-established fruit retailer. To support business development, the company aims to maintain an up-to-date report of its best-selling fruits. Each time a customer makes a purchase, the report should be updated to reflect current sales figures. Furthermore, the report must present fruit sales trends in descending order, ensuring that the highest-selling fruits appear at the top and those with lower sales are listed subsequently.

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

Write JUnit test methods to test your code.

Capabilities tested:

1) Understanding how to use data containers provide by java collection framework
2) Differentiate between how to implement comparison logic in the object versus the logic in client code
3) Understand provided algorithm classes
4) Writing unit test cases

Problem Statement 10:

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

Complete 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_14_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_14_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_14_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”

Capabilities tested:

1) Use Appropriate Java Collection API to solve this problem statement
2) Writing unit test cases

Problem Statement 11:

Word Count

Given a sentence find the number of characters, words and lines in the sentence. Complete the provided method that counts the no. of Characters, Words and Lines in a file and returns the count in a Map.

Input: A sentence

Output: A Map returning the Character count, Word count and Line Count.

Test Case	Input	Output
UTC_15_01	All our dreams can come true, if we have the courage to pursue them	Character count :53 Word count :14 Line count :1
UTC_15_02	Great works are performed, not by strength, but by perseverance	Character count :52 Word count :10

		Line count :2

Problem Statement 13:

Movie and Actors information.

Given data as listed below:

Movie and actor details			
Title	Studio	YEAR	Actors
Skyfall	MGM	2012	Daniel Craig, Javier Bardem, Ralph Fiennes
Schindler's List	Universal Pictures	1993	Liam Neeson, Ben Kingsley
Staying Alive	Paramount Pictures	1983	John Travolta
The Dark Knight Rises	Warner Bros	2012	Christian Bale, Michael Caine, Morgan Freeman
The Taking of Pelham 123	Columbia Pictures	2009	Denzel Washington, John Travolta
Octopussy	United Artists	1983	Roger Moore, Maud Adams
Jurassic Park	Universal Pictures	1993	Sam Neill, Laura Dern, Jeff Goldblum

use appropriate java collection classes for storing and fetching movie/actor details as listed below.

The application has to do the following:

a) Store the above details using appropriate data structures provided by java collection framework.
b) Display movies details and actor names **sorted based on release year**, if there are many movies released in a year they should be sorted alphabetically as shown below:

   Year: 1983
   Movie: Octopussy
   Studio: United Artists
   Actors:
   Roger Moore
   Maud Adams

   Movie: Staying Alive
   Studio: Paramount Pictures
   Actors:
   John Travolta

Year: 1993
Movie: Jurassic Park
Studio: Universal Pictures
Actors:
Sam Neill
Laura Dern
Jeff Goldblum

Movie: Schindler's List
Studio: Universal Pictures
Actors:
Liam Neeson
Ben Kingsley

Year: 2009
Movie: The Taking of Pelham 123
Studio: Columbia Pictures
Actors:
Denzel Washington
John Travolta

etc…

c) Display Movies acted by a given actor. Example is shown below:

i) **Input: John Travolta**
Movies:
Staying Alive
The Taking of Pelham 123

ii) **Input: Ben Kingsley**
Movies:
Schindler's List

Problem Statement 14:

Cryptography

Cryptography: scramble the message.
Cryptography has two disciplines:
– Cryptography: developing codes/ciphers
– Cryptanalysis: breaking codes/ciphers

Symmetric Methods Columnar Transposition Cipher
To encrypt: Write the plaintext in columns of depth k (= key), padding (with *) the end as necessary, read off in rows

Note: Key can be between 1 and (message length -1)

Example if key = 4.
Plaintext: Transposition ciphers are easy!

T	s	i	N	p	s	e	s
r	p	t		h			y
a	o	i	c	e	a	e	!
n	s	o	i	r	r	a	*

Cipher text: Tsinpsesrpt h yaoiceae!nsoirra*

Example if key = 3.
Plaintext: Password Welcome123

P	s	r	W	c	e	3
a	w	d	e	o	1	*
s	o		l	m	2	*

Cipher text: PsrWce3awdeo1*so lm2*

Example if key = 2.
Plaintext: Caeser cipher

C	e	e		i	h	r
a	s	r	c	p	e	*

Cipher text: Cee ihrasrcpe*

Test Case	Input file	Output file
UTC01_01	Transposition ciphers are easy!	Tsinpsesrpt h yaoiceae!nsoirra*
UTC01_02	Password Welcome123	PsrWce3awdeo1*so lm2*
UTC01_03	Caeser cipher	Cee ihrasrcpe*