CRM SFDC
Core Java 8
Lab Book

Document Revision History

Date	Revision No.	Author	Summary of Changes
17-11-2013	1.0	Rathnajothi P	As of updated module content, designed lab book
28-05-2015	2.0	Vinod Satpute	Updated to include new features of Java SE 8, Junit 4 and JAXB 2.0
25-05-2016	3.0	Tanmaya K Acharya	Updated as per the integrated ELT TOC
15-05-2018	4.0	Yogini Naik	Updated as per the TOC of Java Full Stack
1-02-2019	5.0	Yogini Naik	Updated ae per the TOC
Jan 2020	6.0	Ajay P	Updated as per CRM SFDC TOC

Table of Contents

Before you start:

Before your start developing the solutions here are some tips which can make your problem solving easier.

1. Always check if there are any direct API methods available to solve the question easily
2. Use Collection.sort method if you want to sort an arraylist. In case of sorting arrays convert the array and use Collection.sort method to sort it.
3. Converting the numeric data types to string may help to solve some problems for example if you are asked to check if the first digits of two numbers are same convert the two numbers to String and use the charAt() method to check it or if you want to reverse the digits of a number etc.
4. If you are asked to remove the duplicate elements in an array. Convert it to set object. If the array needs to be sorted order go for TreeSet
5. Try to use collection,string and wrapper APIs where ever possible.
6. While using any API methods just go through the other methods in the same API which may help you in solving other problems
7. The Hints provided are just to help the associate solve the problem in the best way. You can use your own algorithm/logic to solve the problem.

Getting Started

Overview

This lab book is a guided tour for learning Core Java version 8. It comprises of assignments to be done. Refer the demos and work out the assignments given by referring the case studies which will expose you to work with Java applications.

Setup Checklist for Core Java

Here is what is expected on your machine in order to work with lab assignment.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 7 or higher.
- Memory: (1GB or more recommended)
- Internet Explorer 9.0 or higher or Google Chrome 43 or higher

Please ensure that the following is done:

- A text editor like Notepad or Eclipse is installed.
- JDK 1.8 or above is installed. (This path is henceforth referred as <java_home>)

Instructions

- For all Naming conventions, refer Appendix A. All lab assignments should adhere to naming conventions.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory java_assignments. For each lab exercise create a directory as lab <lab number>.

Learning More (Bibliography if applicable)

- https://docs.oracle.com/javase/8/docs/
- Java, The Complete Reference; by Herbert Schildt
- Thinking in Java; by Bruce Eckel
- Beginning Java 8 Fundamentals by KishoriSharan

Lab 1: Basics

Exercise 1: Create a class with a method which can calculate the sum of first n natural numbers which are divisible by 3 or 5.

Method Name	calculateSum
Method Description	Calculate Sum
Argument	int n
Return Type	int-sum
Logic	Calculate the sum of first n natural numbers which are divisible by 3 or 5.

Exercise 2: Create a class with a method to find the difference between the sum of the squares and the square of the sum of the first n natural numbers.

Method Name	calculateDifference
Method Description	Calculate the difference
Argument	int n
Return Type	int - Sum
Logic	Find the difference between the sum of the squares of the first n natural numbers and the square of their sum. For Example if n is 10,you have to find (1^2+2^2+3^2+….9^2+10^2)-(1+2+3+4+5…+9+10)^2

Exercise 3: Create a method to check if a number is an increasing number

Method Name	checkNumber
Method Description	Check if a number is an increasing number
Argument	int number
Return Type	boolean
Logic	A number is said to be an increasing number if no digit is exceeded by the digit to its left. For Example : 134468 is an increasing number

Exercise 4: Create a method to check if a number is a power of two or not

Method Name	checkNumber
Method Description	Checks if the entered number is a power of two or not
Argument	int n
Return Type	boolean
Logic	Check if the input is a power of two. Ex: 8 is a power of 2

Lab 2: Inheritance and Polymorphism

Using an inheritance hierarchy, design a Java program to model items at a library (books, journal articles, videos and CDs.) Have an abstract superclass called Item and include common information that the library must have for every item (such as unique identification number, title, and number of copies). No actual objects of type Item will be created - each actual item will be an object of a (non-abstract) subclass. Place item-type-specific behavior in subclasses (such as a video's year of release, a CD's musical genre, or a book's author). More in detail:

1. Implement an abstract superclass called Item and define all common operations on this class (constructors, getters, setters, equals, toString, print, checkIn, checkOut, addItem, etc). Have private data for: identification number, title, and number of copies.

2. Implement an abstract subclass of Item named WrittenItem and define all common operations on this class. Added private data for author.

3. Implement 2 subclasses of WrittenItem: Book and JournalPaper.

    3.1. Class Book: no new private data. When needed, override/overload methods from the superclass.
    3.2. Class JournalPaper: added private data for year published. When needed, override/overload methods from the superclass.

4. Implement another abstract subclass of Item named MediaItem and define all common operations on this class. Added private data for runtime (integer).

5. Implement 2 subclasses of MediaItem: Video and CD.

    5.1. Class Video: added private data for director, genre and year released. When needed, override/overload methods from the superclass.
    5.2. Class CD: added private data for artist and genre. When needed, override/overload methods from the superclass.

Write the definitions of these classes and a client program (your choice!) showing them in use.

Lab 3: Assignments

Exercise 1: Create a method which accepts an array of integer elements and return the second smallest element in the array

Method Name	getSecondSmallest
Method Description	Get the second smallest element in the array
Argument	int[]
Return Type	int
Logic	Sort the array and return the second smallest element in the array

Exercise 2: Create a method that can accept an array of String objects and sort in alphabetical order. The elements in the left half should be completely in uppercase and the elements in the right half should be completely in lower case. Return the resulting array.
Note: If there are odd number of String objects, then (n/2) +1 elements should be in UPPPERCASE
Exercise 3: Create a method which accepts an integer array, reverse the numbers in the array and returns the resulting array in sorted order

Method Name	getSorted
Method Description	Return the resulting array after reversing the numbers and sorting it
Argument	int []
Return Type	int
Logic	Accept and integer array, reverse the numbers in the array, sort it and return the resulting array. Hint Convert the numbers to String to reverse it

Exercise 4: Create a method that accepts a character array and count the number of times each character is present in the array.

Lab 4: Operators

Exercise 1: Create a method to find the sum of the cubes of the digits of an n digit number

Lab 5: Flow control and Exception Handling

Exercise 1: Write a java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green with radio buttons. On entering the choice, an appropriate message with “stop” or “ready” or “go” should appear in the console .Initially there is no message shown.

Exercise 2: The Fibonacci sequence is defined by the following rule. The first 2 values in the sequence are 1, 1. Every subsequent value is the sum of the 2 values preceding it. Write a Java program that uses both recursive and nonrecursive functions to print the nth value of the Fibonacci sequence?

Exercise 3: Write a Java program that prompts the user for an integer and then prints out all the prime numbers up to that Integer?

Exercise 4: Write a Java Program to validate the full name of an employee. Create and throw a user defined exception if firstName and lastName is blank.

Exercise 5: Validate the age of a person and display proper message by using user defined exception. Age of a person should be above 15.

Exercise 6: Create an Exception class named as “EmployeeException”(User defined Exception) in a package named as “com.cg.eis.exception” and throw an exception if salary of an employee is below than 3000. Use Exception Handling mechanism to handle exception properly.

Lab 6: Strings, I/O Formatting and Parsing

Exercise 1: Write a Java program that reads a line of integers and then displays each integer and the sum of all integers. (Use StringTokenizer class)?

Exercise 2: Write a Java program that reads a file and displays the file on the screen, with a line number before each line?

Exercise 3: Write a Java program that displays the number of characters, lines and words in a text?

Exercise 4: Write a Java program that reads on file name from the user, then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file and the length of the file in bytes?

Exercise 5: Create a method that accepts a String and checks if it is a positive string. A string is considered a positive string, if on moving from left to right each character in the String comes after the previous characters in the Alphabetical order. For Example: ANT is a positive String (Since T comes after N and N comes after A). The method should return true if the entered string is positive.

Exercise 6: Create a method to accept date and print the duration in days, months and years with regards to current system date.

Exercise 7: You are asked to create an application for registering the details of jobseeker. The requirement is:
Username should always end with _job and there should be at least minimum of 8 characters to the left of _job. Write a function to validate the same. Return true in case the validation is passed. In case of validation failure return false.

Lab 7: Collection and Generics

Exercise 1: Create a method which accepts a hash map and return the values of the map in sorted order as a List.

Method Name	getValues
Method Description	Get the values of a map in sorted order
Argument	HashMap
Return Type	List
Logic	Return the values of a hash map in sorted order

Exercise 2: Create a method that accepts a character array and count the number of times each character is present in the array. Add how many times each character is present to a hash map with the character as key and the repetitions count as value

Method Name	countCharacter
Method Description	Count the number of occurrence of each character in a Character array
Argument	char[]
Return Type	map

Logic	Count the number of times each character appears in the array. Add the details into a hash map with character as key and count as value. Example: {'A','P','P','L','E'} Output: Will be hashmap with the following contents{'A':1,'P':2,'L':1,'E':1}

Exercise 3: Create a method which accepts an array of numbers and returns the numbers and their squares in HashMap

Method Name	getSquares
Method Description	Accepts a list of numbers and return their squares
Argument	int[]
Return Type	Map
Logic	Iterate through the list, find the square of each number and add the elements to a map object with the number as the key and the square as the value

Lab 8: Layered Architecture

Refer the case study and create an application for that requirement by creating packages and classes as given below:

Case Study:

1. **Employee Medical Insurance Scheme:**

   - By default, all employees in an organization will be assigned with a medical insurance scheme based on the salary range and designation of the employee. Refer the below given table to find the eligible insurance scheme specific to an employee.

Salary	Designation	Insurance scheme
>5000 and < 20000	System Associate	Scheme C
>=20000 and <40000	Programmer	Scheme B
>=40000	Manager	Scheme A
<5000	Clerk	No Scheme

a) com.cg.eis.bean

In this package, create “Employee” class with different attributes such as id, name, salary, designation, insuranceScheme.

b) com.cg.eis.service

This package will contain code for services offered in Employee Insurance System. The service class will have one EmployeeService Interface and its corresponding implementation class.

c) com.cg.eis.pl

This package will contain code for getting input from user, produce expected output to the user and invoke services offered by the system.

The services offered by this application currently are:

i) Get employee details from user.

ii) Find the insurance scheme for an employee based on salary and designation.

iii) Display all the details of an employee.

Use overrides annotation for the overridden methods available in a derived class of an interface of all the assignments.

