**Lab 1:**

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: Inheritanceand 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 methodwhich 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 10: Lambda Expressions and Stream API

Exercise 1: Write a lambda expression which accepts x and y numbers and return xy.

Exercise 2: Write a method that uses lambda expression to format a given string, where a space is inserted between each character of string. For ex., if input is “CG”, then expected output is “C G”.

Exercise 3: Write a method that uses lambda expression to accept username and password and return true or false. (**Hint:** Use any custom values for username and password for authentication)

Exercise 4: Write a class with main method to demonstrate instance creation using method reference. (**Hint:** Create any simple class with attributes and getters and setters)

Exercise 5: Write a method to calculate factorial of a number. Test this method using method reference feature.

**Case Study for Steam API:**

Refer the classes given below to represent employees and their departments.



Class Diagram used for Stream API

Also refer an EmployeeRepository class which is used to create and populate employee’s collection with sample data.



Create an EmployeeService class which queries on collections provided by EmployeeRepository class for following requirements. Create separate method for each requirement. (**Note:** Each requirement stated below must be attempted by using lambda expressions/stream API. It’s mandatory to solve at least 5 questions from following set. However, it is recommended to solve all questions to understand stream API thoroughly).

Find out the sum of salary of all employees.

List out department names and count of employees in each department.

Find out the senior most employee of an organization.

List employee name and duration of their service in months and days.

Find out employees without department.

Find out department without employees.

Find departments with highest count of employees.

List employee name, hire date and day of week on which employee has started.

Revise exercise 10.13 to list employee name, hire date and day of week for employee started on Friday. (Hint: Accept the day name for e.g. FRIDAY and list all employees joined on Friday)

List employee’s names and name of manager to whom he/she reports. Create a report in format “employee name reports to manager name”.

List employee name, salary and salary increased by 15%.

Find employees who didn’t report to anyone (**Hint:** Employees without manager)

Create a method to accept first name and last name of manager to print name of all his/her subordinates.

Sort employees by their

* Employee id
* Department id
* First name

**Lab 11: 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 |

1. **com.cgi.eis.bean**

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

1. **com.cgi.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.

1. **com.cgi.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:

1. Get employee details from user.
2. Find the insurance scheme for an employee based on salary and designation.
3. 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.