LAB WORKBOOK

CSIT725 Core Java

LABORATORY WORKBOOK

STUDENT NAME	
REG. NO	
YEAR	
SEMESTER	
SECTION	
FACULTY	

Table of contents

Organization of the STUDENT LAB WORKBOOK

The laboratory framework includes a creative element but shifts the time-intensive aspects outside of the Two-Hour closed laboratory period. Within this structure, each laboratory includes three parts: Prelab, In-lab, and Post-lab.

a. Pre-Lab

The Prelab exercise is a homework assignment that links the lecture with the laboratory period - typically takes 2 hours to complete. The goal is to synthesize the information they learn in lecture with material from their textbook to produce a working piece of software. Prelab Students attending a two-hour closed laboratory are expected to make a good-faith effort to complete the Prelab exercise before coming to the lab. Their work need not be perfect, but their effort must be real (roughly 80 percent correct).

b. In-Lab

The In-lab section takes place during the actual laboratory period. The First hour of the laboratory period can be used to resolve any problems the students might have experienced in completing the Prelab exercises. The intent is to give constructive feedback so that students leave the lab with working Prelab software - a significant accomplishment on their part. During the second hour, students complete the In-lab exercise to reinforce the concepts learned in the Prelab. Students leave the lab having received feedback on their Prelab and In-lab work.

c. Post-Lab

The last phase of each laboratory is a homework assignment that is done following the laboratory period. In the Post-lab, students analyze the efficiency or utility of a given system call. Each Post-lab exercise should take roughly 120 minutes to complete.

COURSE CODE: CSIT 725

Core Java

Lab 1:

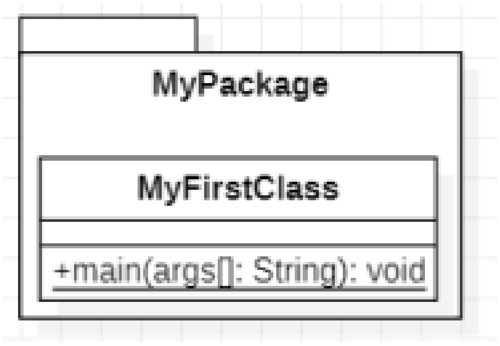
Date of the Session: ___/___/___ Time of the Session: _____to______

Prerequisite: Java fundamentals- Operators, Conditions, loops and output statement

Pre-Lab Task:

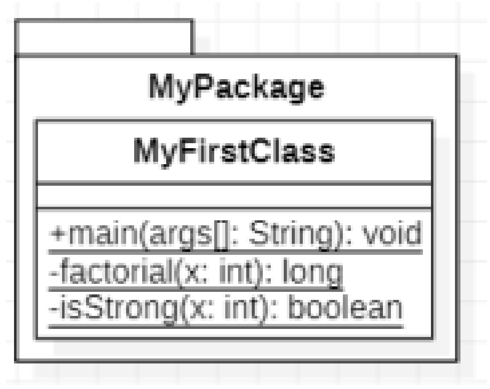
1. Write a Java Program to print the “Hello World” statement.

2. Write a Java program to compute the factorial of a number (assume hardcoded user input) in main method

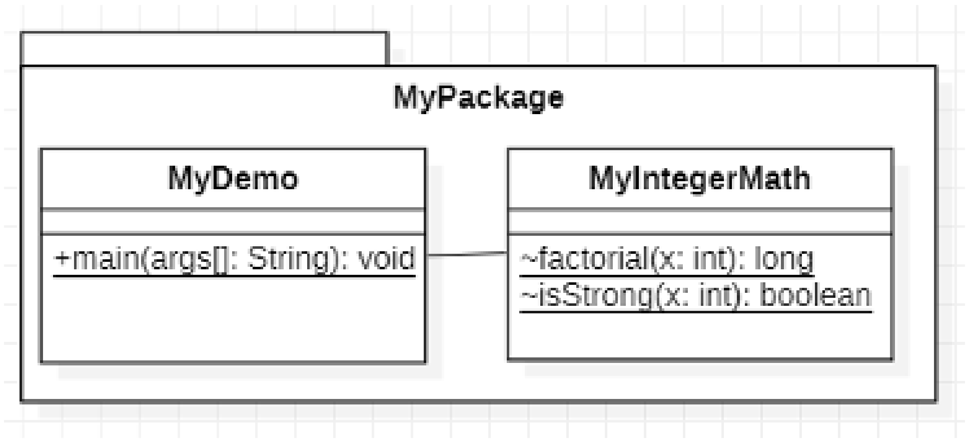


In-Lab Task:

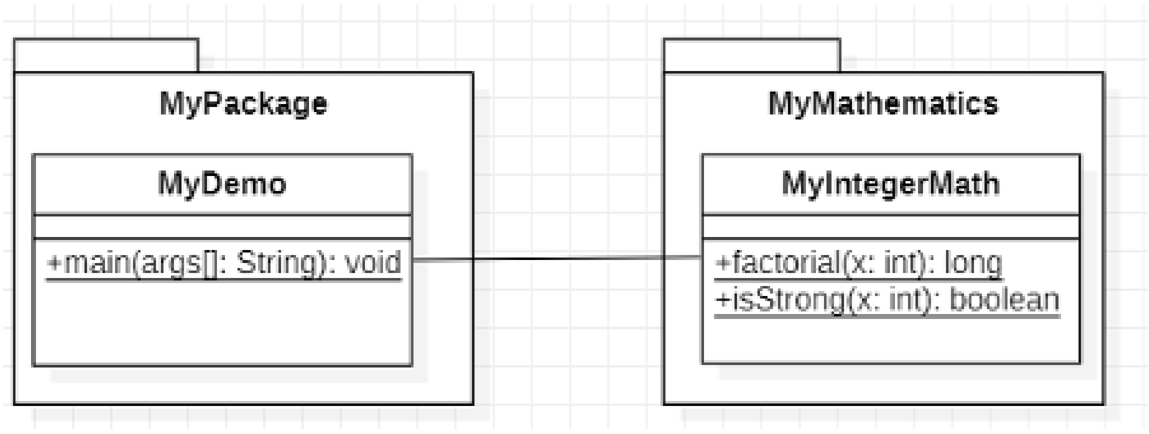
1. Develop Java code with 2 methods in MyFirstClass a) factorial () b) isStrong() *Strong number* is a special number whose sum of factorial of digits is equal to the original number. For example: 145 is strong number. Since, 1! + 4! + 5! = 145. The method expects an integer as argument and then returns true if the number is strong, otherwise returns false.



2. Rework of Q1, modularizing to a class level as follows:



3. Rework on Q3, modularizing to package level.



Post-Lab Task:

1. Modify MyIntegerMath class, such that it contains the following methods

1) countDigits() 2)isArmstrong().

countDigits() expects an integer as argument and returns the number of digits in it. isArmstrong() expects an integer as argument and returns true if it is an Armstrong number otherwise returns false.

A positive integer of n digits is called an Armstrong number of order n (order is number of digits) if **abcd... = pow(a,n) + pow(b,n) + pow(c,n) + pow(d,n) +** ....

Draw the class diagrams:
a) modularized to class level
b) modularized to package level

2. Develop java code for the above designs a) and b).

Lab 2:

Date of the Session: ___/___/___ Time of the Session: _____to______

Prerequisite: Syntax to define class, Need for class as a template, Command line arguments and console input

Pre-Lab Task:

1. Design a class named Stock that contains:
   a) A string data field named symbol for the stock's symbol.
   b) A string data field named name for the stock's name.
   c) A double data field named previousClosingPrice that stores the stock price for the previous day.
   d) A double data field named currentPrice that stores the stock price for the current time.
   e) A method named getChangePercent() that returns the percentage changed from previousClosingPrice to currentPrice.
   Use appropriate access specifiers for **instance variables** and **instance methods**. Draw the class diagram.
   (Hint: Modularize to package level)

2. Write menu driven main () method in Demo class to perform a) Linear Search
   b) Binary Search
   c) Bubble Sort

   (Hint: Use static methods in same class.)

   The program must read a set of integers through command line arguments and use console to read the key.

In-Lab Task:

1. Design a class named QuadraticEquation for a quadratic equation $ax^2 + bx + c = 0$. The class contains:
   a) Private data fields a, b, and c that represent three coefficients.
   b) Three getter methods for a, b, and c.
   c) A method named getDiscriminant() that returns the discriminant, which is $b^2 - 4ac$.
   d) The methods named getRoot1() and getRoot2() for returning two roots of the equation.

   r1 and r2 = $\dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

2.Define a class Student with the following attributes under private access and methods under public access.

a. Name b) ID c) gender d) department

i. Define setter methods such that name but not have any special characters and Digits.

ii. ID must be a positive 9-digit value

iii. Gender must be either M/F

iv. Department must be either BT/CE/CSE/ECE/EEE/ECS/ME/PE v. Define toString() method

vi. In the main method define two Student objects and prints the details of the students in the following format.

```
ID: 190030000
Name: ABC
Gender: M
Department: CSE
```

(Hint: Read the data from console)

Post-Lab Task:

1. Write a program that randomly generates an array of 100,000 integers and a key. Estimate the execution time of invoking the linearSearch() method in best, average and worst cases.

   You can use the following code template to obtain the execution time:

   a. long startTime = System.currentTimeMillis();

   b. perform the task;

   c. long endTime = System.currentTimeMillis();

   d. long executionTime = endTime - startTime;

2. Develop a java program to that reads the number of rows and columns through command-line, reads the set of elements in a 2D array from console. The menu driven program must perform the following operations:

   a) Sum of all elements

   b) Print the data in matrix form

   c) Print the elements of principal diagonal

   d) Print the sum of elements in Principal diagonal

Lab 3:

Date of the Session: ___/___/___ Time of the Session: _____to______ Prerequisite: I/O through files and GUI

Pre-Lab Task:

1. Alex has a file which has 2 integers. The task is to add these integers and print the result. Write java program to perform this task.

2. Write a java program that reads Student ID, Name, age, gender and prints the details on User Interface.
   (Hint: Use JOptionPane of javax.swing package)

In-Lab Task:

1. Chandu has 2 files, Names.txt and Address.txt. Names.txt has 2 columns ID and student names and Dept.txt has 2 columns ID and Address. Fortunately, the IDs in both files are same and sorted. The task is to read data from both files and print the student data in the following format:
   ID: Name: Address:
   Print one student details per line on the console.

2. Develop a java program using swing package. The program must display a student registration page, which reads ID, Name, Gender (Use Radio Buttons) and department (Use Drop down selection) and two buttons Submit and Reset. When the user clicks Submit, then validate the data, use JOptionPane to alert if any data is missing or entered wrong, otherwise display the data submitted back on JOptionPane. The reset button clears all the data.

Post-Lab Task:

1. Develop a simple calculator to perform basic arithmetic on 2 integers using GUI.

Lab 4:

Date of the Session: ___/___/___ Time of the Session: _____to______ <u>Prerequisite:</u>

Constructors and Overloading, Chaining and this

<u>Pre-Lab Task:</u>

1. Design a class named MyInteger. The class contains:
✔ An int data field named value that stores the int value represented by this object.
✔ A constructor that creates a MyInteger object for the specified int value. ✔ A getter method that returns the int value.
✔ The methods isEven(), isOdd(), and isPrime() that return true if the value in this object is even, odd, or prime, respectively.
✔ The static methods isEven(int), isOdd(int), and isPrime(int) that return true if the specified value is even, odd, or prime, respectively.
✔ The static methods isEven(MyInteger), isOdd(MyInteger), and isPrime(MyInteger) that return true if the specified value is even, odd, or prime, respectively. ✔ The methods equals(int) and equals(MyInteger) that return true if the value in this object is equal to the specified value.
✔ A static method parseInt(char[]) that converts an array of numeric characters to an int value.
✔ A static method parseInt(String) that converts a string into an int value.

Draw the class diagram for the class and then implement the class. Write a client program that tests all methods in the class.

2. Enhance the above my adding a method factorial () which expects an integer argument and returns the factorial value as BigInteger Wrapper class.

<u>In-Lab Task:</u>

1. Design a class named Account that contains:

• A private int data field named id for the account (default 0).
• A private double data field named balance for the account (default 0). • A private double data field named annualInterestRate that stores the current interest rate (default 0). Assume all accounts have the same interest rate.
• A no-arg constructor that creates a default account.
• A constructor that creates an account with the specified id and initial balance.
• The accessor and mutator methods for id, balance, and annualInterestRate.
• Mutators return Boolean (If all the fields must be +ve, return true, else false)
• A method named withdraw that withdraws a specified amount from the account. • A method named deposit that deposits a specified amount to the account.

Draw the UML diagram for the class and then implement the class.
(Hint: The method getMonthlyInterest() is to return monthly interest, not the interest rate
Monthly interest is balance * monthlyInterestRate.

monthlyInterestRate is annualInterestRate / 12.)

Note that annualInterestRate is a percentage,e.g., like 4.5%. You need to divide it by 100.) Write a test program that creates an Account object with an account ID of 1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the withdraw method to withdraw $2,500, use the deposit method to deposit $3,000, and print the balance, the monthly interest.

2. Design a class named MyPoint to represent a point with x- and y-coordinates. The class contains:

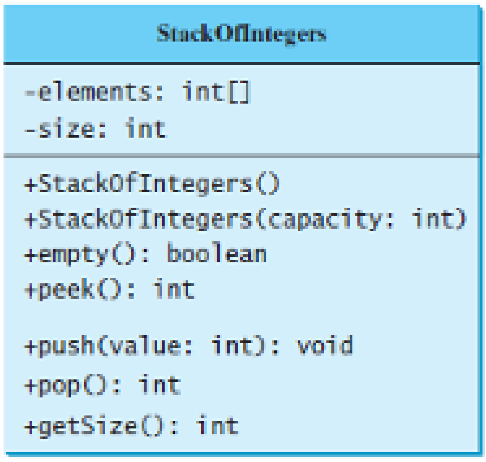
The data fields x and y that represent the coordinates with getter methods.

■ A no-arg constructor that creates a point (0, 0).
■ A constructor that constructs a point with specified coordinates.
■ A method named distance that returns the distance from this point to a specified point of the MyPoint type.
■ A method named distance that returns the distance from this point to another point with specified x- and y-coordinates.
Draw the UML diagram for the class and then implement the class. Write a test program that creates the two points (0, 0) and (10, 30.5) and displays the distance between them.

Post-Lab Task:

1. Design Use the Account class that simulates an ATM machine. Create ten accounts in an array with id 1, . . . , 10, and initial balance $100. The system prompts the user to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, the main menu is displayed as shown in the sample run. You can enter a choice 1 for viewing the current balance, 2 for withdrawing money, 3 for depositing money, and 4 for exiting the main menu. Once you exit, the system will prompt for an id again. Thus, once the system starts, it will stop when id is 0.

2. Implement the following



```
StackOfIntegers

-elements: int[]
-size: int

+StackOfIntegers()
+StackOfIntegers(capacity: int)
+empty(): boolean
+peek(): int

+push(value: int): void
+pop(): int
+getSize(): int
```

In the no-args constructor, size must be initialized to 10 and initialize the array with the same.

Lab 5:

Date of the Session: ___/___/___ Time of the Session: _____to______

<u>Prerequisite:</u> Objects as data members, Menu driven programs to perform operations on array of objects

<u>Pre-Lab Task:</u>

1. Design a class named **StopWatch**. The class contains:
■ Private data fields **startTime** and **endTime** with getter methods.
■ A no-arg constructor that initializes **startTime** with the current time. ■ A method named **start()** that sets the **startTime** to the current time. ■ A method named **stop()** that sets the **endTime** to the current time.
■ A method named **getElapsedTime()** that returns the elapsed time for the stopwatch in milliseconds.
Draw the UML diagram for the class and then implement the class.

2. A Restaurant serves dishes of three types: Veg, Non-Veg and Egg represented by green, red and brown color codes respectively. Help them out by writing a menu driven program to display the various dishes depending on the type of food the customer opts for.

<u>In-Lab Task:</u>

1. A customer of a bank wants to withdraw/deposit money from his account. There are 3 ATMs in his town. Help him out by writing a program such that his balance will be updated after a transaction in any of the ATMs. (Use 'Account' as Singleton Class and it should be Early Instantiated)

2. A company wants to digitalize their manual records of the employee details (Employee ID, Employee name, Employee Department). If all the three fields are given, use the given details. If not, use the default values.
(Use Constructor Overloading)
Default values are:
ID: 0
Name: #
Department: #.
Write toString () method, mutators and accessors.
Write a menu driven main () method to perform the following operations:
Create new Employee record
Update name based on ID
Print All Employees
Print Department Specific employees

Post-Lab Task:

1. Use the student class define in Lab 2 and enhance the main method such that we can store data of 10 students in an array and perform the following operations as a menu driven program.
a) Create a new student
b) Print details of all students
c) Print details based on ID
d) Modify student name based on ID
e) Remove a student based on ID

2. Enhance the design of In-lab Q2, such that the date of joining is one attribute of Employee and print the details of all employees who joined in 2019.

Lab 6: Strings

Date of the Session: ___/___/___ Time of the Session: _____to______ <u>Prerequisite:</u>

String, StringBuffer, StringTokenizer and Date

<u>Pre-Lab Task:</u>

1. Write a menu driven program to illustrate the following operations on String Class a) charAt()
b) length()
c) indexOf() -all overloaded methods
d) lastIndexOf() -all overloaded methods
e) substring() - all overloaded methods
f) valueOf() - all overloaded methods

2. Write a menu driven program to illustrate the following operations on StringBuffer Class
a) charAt()
b) append()
c) capacity()
d) length()
e) delete()
f) deleteCharAt()
g) insert()
h) reverse()
i) replace()

<u>In-Lab Task:</u>

1. Write a program that sets up a String variable containing a paragraph of text of your choice from file. Extract the words from the text and sort them into alphabetical order. Display the sorted list of words. You could use a simple sorting method called the bubble sort.

2. Define an array of ten String elements each containing an arbitrary string of the form "month/day/year"; for example,"10/29/99" or "12/5/01". Analyze each element in the array and output the date represented in the form 29th October 1999

<u>Post-Lab Task:</u>

1. Write a program that will reverse the sequence of letters in each word of your chosen paragraph from Exercise 3. For instance, "To be or not to be." would become "oT eb ro ton ot eb."

Lab 07:

Date of the Session: ___/___/___ Time of the Session: _____to______ <u>Prerequisite:</u>

Inheritance, method overloading

<u>Pre-Lab Task:</u>

1. Develop a program that creates a generic Shape class with attributes fillColor, borderColor, fill (Boolean type) and border width. The classes Rectangle, Circle must inherit Shape. Rectangle has length and width as attributes and circle with radius as attribute. Also add the corresponding getters and setters, toString() in each of the classes. Use appropriate access specifiers. In the main () method of Demo class, create objects and access the methods. Draw the class diagram.

2. Enhance the Pre-lab Q1 hierarchy such that Shape is the general class, inherited by TwoDShape and ThreeDShape class, and Rectangle, Circle inherits the TwoDShape, Cuboid and Sphere extend the ThreeDshape.