# K.S. RANGASAMY COLLEGE OF TECHNOLOGY TIRUCHENGODE-637 215

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

****

# LAB RECORD

**50 CS 4P1- JAVA PROGRAMMING LABORATORY BATCH:2022-2025**

**Name: Register Number: Year/Sem:**

# K.S. RANGASAMY COLLEGE OF TECHNOLOGY,

# TIRUCHENGODE-637 215

# DEPARTMENT OF COMPUTER SCIENCE AND

# ENGINEERING

****

|  |  |
| --- | --- |
| **REGISTER NUMBER** |  |

Certified that this is the Bonafide record for the practical work done by Selvan / Selvi of II year IV Semester for the course 50CS4P1-JAVA PROGRAMMING LABORATORY during the academic year 2022-2023.

**STAFF - IN- CHARGE HEAD OF THE DEPARTMENT**

**Submitted for the Practical Examination held on ………………….**

**INTERNAL EXAMINER SUBJECT EXPERT**

**K.S. RANGASAMY COLLEGE OF TECHNOLOGY, TIRUCHENGODE-637 215**

**(Autonomous)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING VISION AND MISSION OF THE INSTITUTION**

## VISION

To produce the most competent Scientists, Engineers, Technologists, Entrepreneurs, Managers and Researchers through quality Education.

## MISSION

To achieve academic excellence in Science, Engineering, Technology, Management and Research through objective and innovative teaching methods; dedicated and duty conscious faculty; continual and consistent updation of facilities; welfare and quality improvement of the faculty and a system of continual process improvement.

**VISION AND MISSION OF CSE DEPARTMENT**

### VISION

To produce competent software professionals, academicians and researchers through Quality Education.

### MISSION

* To produce competent software developers, system designers and network programmers.
* To keep abreast of the latest developments and technological transformations in computer science and engineering for social benefits.

### PROGRAM EDUCATIONAL OBJECTIVES (PEOS):

A graduate of the Computer Science and Engineering Program should:

|  |  |
| --- | --- |
| **PEO1** | Graduates will provide effective solutions for software and hardware industries by applying the concepts of basic science and engineering fundamentals. |
| **PEO2** | Graduates will be professionally competent and successful in their career through life-long learning. |
| **PEO3** | Graduates will contribute individually or as member of a team in handling projects and  demonstrate social responsibility and professional ethics |

### PROGRAM OUTCOMES (POS):

|  |  |
| --- | --- |
| **PO1** | **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| **PO2** | **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. |
| **PO3** | **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. |
| **PO4** | **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions**.** |
| **PO5** | **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and  modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. |
| **PO6** | **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. |
| **PO7** | **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. |
| **PO8** | **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| **PO9** | **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings**.** |
| **PO10** | **Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. |
| **PO11** | **Project management and finance:** Demonstrate knowledge understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. |
| **PO12** | **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. |

**PROGRAM SPECIFIC OUTCOMES(PSOs):**

|  |  |
| --- | --- |
| **PSO1** | **Software Project Management:** Apply standard Software Engineering practices and strategies in software project development using open-source programming environment to deliver a quality product for business success. |
| **PSO2** | **Data Analytics:** Analyze and Interpret data by adapting advanced data analytics models for decision making in Complex Problems and facilitate inter disciplinary research. |

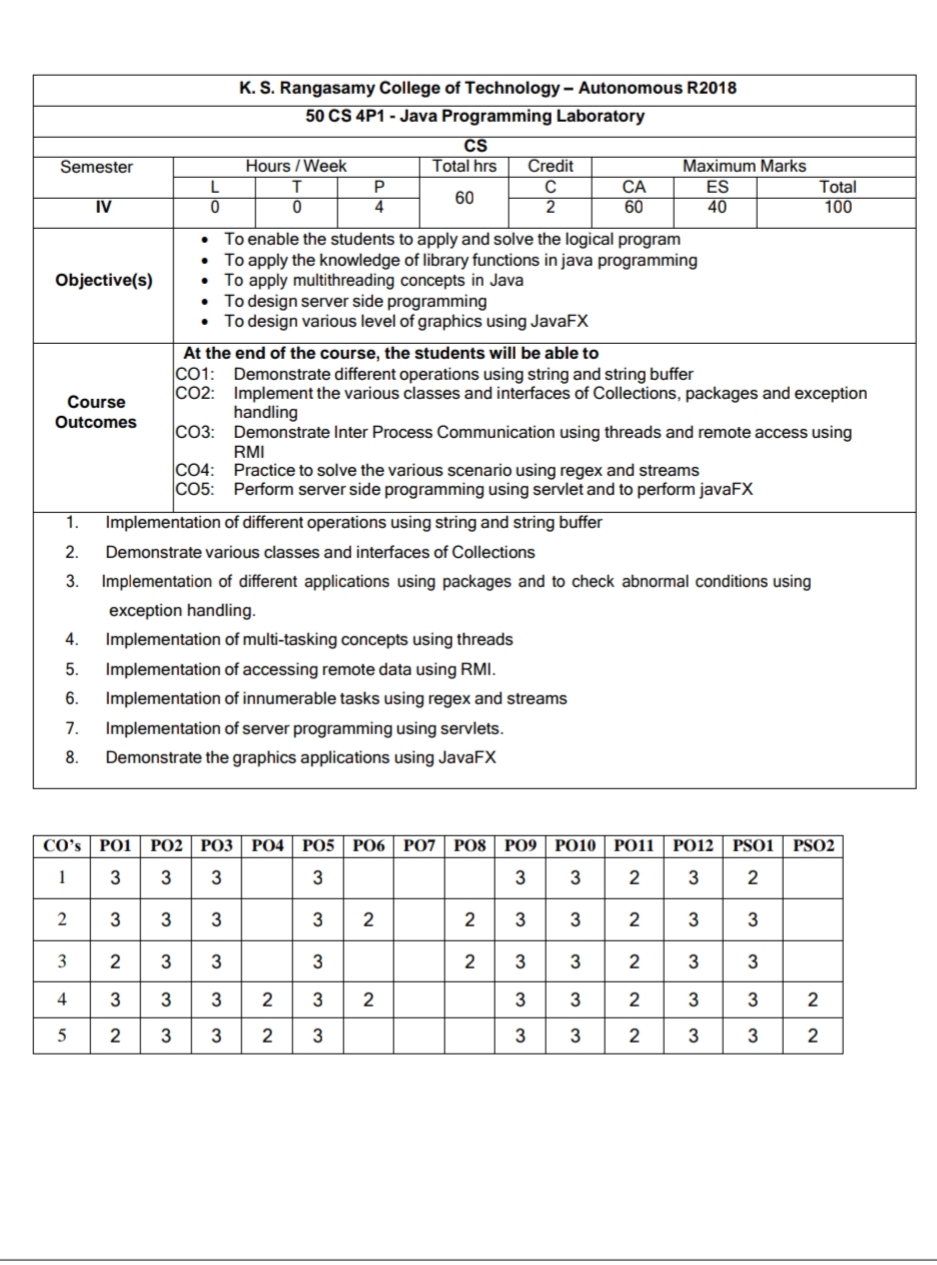
### K.S. RANGASAMY COLLEGE OF TECHNOLOGY

### (Autonomous)

### TIRUCHENGODE-637 215

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GENERAL LABORATORY INSTRUCTIONS**

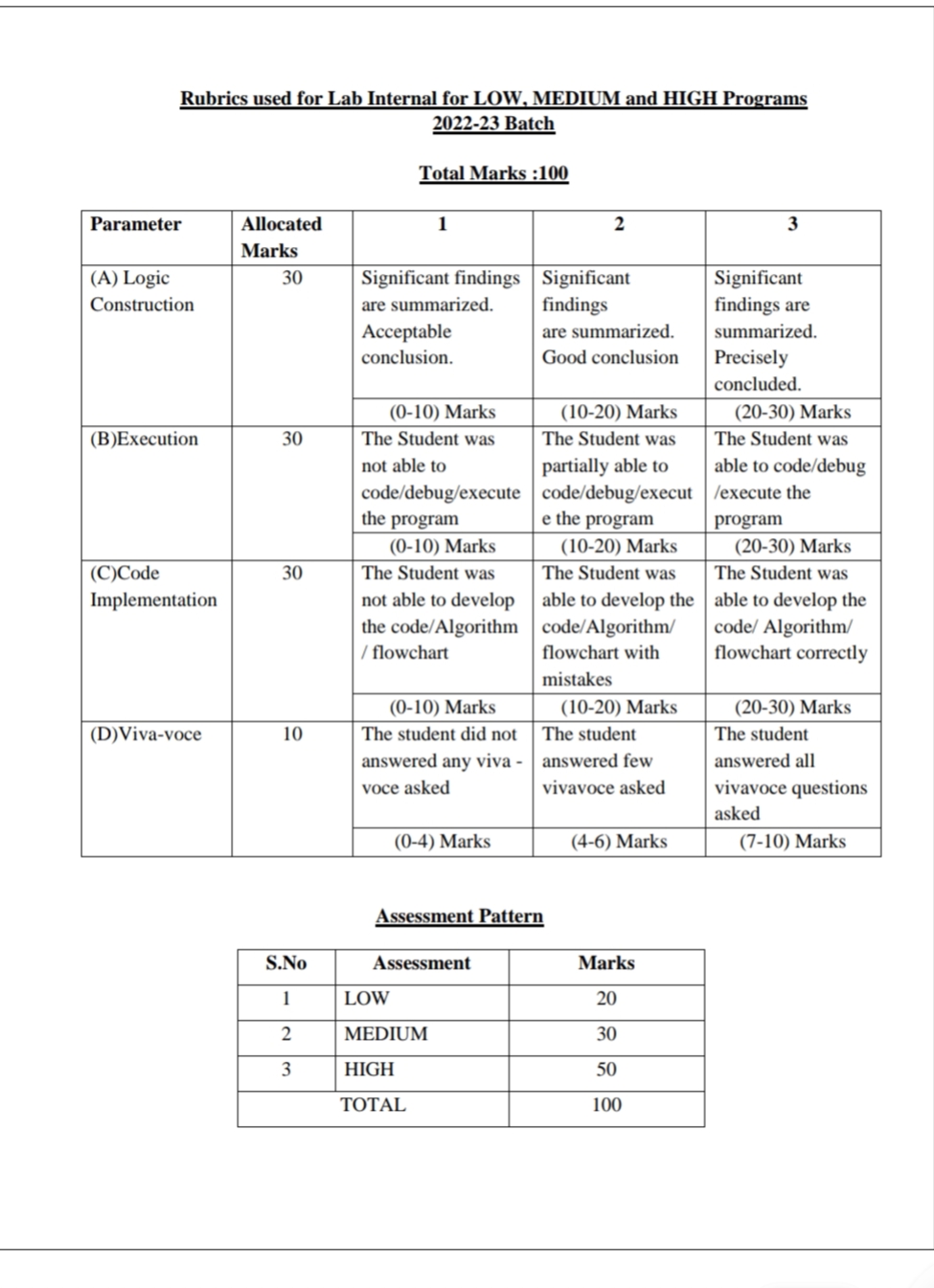
|  |  |
| --- | --- |
| 1 | Students are advised to come to the laboratory at least 5 minutes before (to the starting time),  those who come after 5 minutes will not be allowed into the lab. |
| 2 | Plan your task properly much before to the commencement, come prepared to the lab. |
| 3 | Student should enter into the laboratory with:  Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.  Laboratory Record updated up to the last session exercises in the lab.  Proper Dress code and Identity card. |
| 4 | Sign in the laboratory login register, write the TIME-IN, and occupy the computer system  allotted to you by the faculty. |
| 5 | Execute your task in the laboratory, and record the results / output in the lab observation note  book, and get certified by the concerned faculty. |
| 6 | All the students should be polite and cooperative with the laboratory staff, must maintain the  discipline and decency in the laboratory. |
| 7 | Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab  sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment. |
| 8 | Students must take the permission of the faculty in case of any urgency to go out; if anybody  found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately. |
| 9 | Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is  kept properly. |

****

### INDEX

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ex.No** | **Date** | **Name of the Exercise** | **Page**  **No** | **Marks**  **Awarded** | **Staff**  **Signature** |
| **1** | **25.01.23** | **Implementation of different operations using string and string buffer** | **15** |  |  |
| **2** | **01.02.23** | **Demonstrate various classes and interfaces of Collections** | **21** |  |  |
| **3** | **14.02.23** | **Implementation of different applications using packages and to check abnormal**  **conditions using exception handling.** | **30** |  |  |
| **4** | **01.03.23** | **Implementation of multi-tasking concepts using threads** | **41** |  |  |
| **5** | **15.03.23** | **Implementation of accessing remote data using RMI.** | **49** |  |  |
| **6** | **11.04.23** | **Implementation of innumerable tasks using regex and streams** | **52** |  |  |
| **7** | **19.04.23** | **Implementation of server programming using servlets.** | **55** |  |  |
| **8** | **10.05.23** | **Demonstrate the graphics applications using JavaFX** | **59** |  |  |
|  |  | **Mini Project:** |  |  |  |
| **Average** | | | |  |  |

**STAFF SIGNATURE**

****

**INTRODUCTION**

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

Java is guaranteed to be **Write Once, Run Anywhere.**

Java is:

* Object Oriented: In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
* Platform independent: Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on which ever platform it is being run.
* **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP Java would be easy to master.
* **Secure:** With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
* **Architectural-neutral**: Java compiler generates an architecture-neutral object file format which makes the compiled code to be executable on many processors, with the presence of Java runtime system.
* **Portable:** Being architectural-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary which is a POSIX subset.
* **Robust:** Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
* **Multithreaded:** With Java's multithreaded feature it is possible to write programs that can do many tasks simultaneously. This design feature allows developers to construct smoothly running interactive applications.
* **Interpreted:** Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light weight process.
* **High Performance:** With the use of Just-In-Time compilers, Java enables high performance.
* **Distributed:** Java is designed for the distributed environment of the internet.
* **Dynamic:** Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

**Creating a simple Java Program**

Hello World example:

class HelloWorld {

public static void main (String args[]) {

System.out.println("Hello World! ");

}}

This program has two main parts:

* All the program is enclosed in a class definition—here, a class called Hello World.
* The body of the program (here, just the one line) is contained in a method (function) called main(). In Java applications, as in a C or C++ program, main() is the first method (function)that is run when the program is executed.

**Compiling the above program:**

In Sun's JDK, the Java compiler is called javac.

***javacHelloWorld.java***

When the program compiles without errors, a file called HelloWorld.class is created, in the same directory as the source file. This is the Java bytecode file. Then run that byte code file using the Java interpreter. In the JDK, the Java interpreter is called simply java.

***java HelloWorld***

If the program was typed and compiled correctly, the output will be:

"Hello World!"

**Coding Guidelines**

In Java, the coding guidelines are the set of rules that are followed by the developer during application development. These guidelines provide readability to other developer and user who are dealing with the project. The guidelines must be followed because an application is not developed by a single programmer. Some major coding guidelines include the following:

**Naming Conventions**

* The class and interface name must be noun, and the first letter of each internal word should be capitalized.
* The method name must be verb in mixed case, each first letter should be in lower case with the first letter of each internal word should be capitalized.
* All the constants should be in capital letters.
* The variable name must be a meaningful letter or word.

**Curly Braces**

The use of curly braces is most important in programming. It defines the body of the class, methods, and loop.

**Indentation**

The indentation should be 4 spaces. By pressing the Tab key, we get exactly 8-spaces. Indentation can be achieved by the space character and the tab characters

**White Spaces** It also plays a vital role in readability of the program. Any mathematical operations (+, -, \*, %, etc.) should be surrounded by a space character.

**Comments**

Comments in programming, is very important because it enhance the readability of the program. It contains the relevant information regarding program. Note that comments are ignored by the compiler.

**Java Lib**

Java Standard Library is one of the most popular and used libraries, which contains a list of libraries to make work easier. These libraries are called at run time by JVM(Java Virtual Machine). It provides the following libraries:

* We cannot write any program in Java without String, Enum, Double, etc. The lang library provides everything to us for writing code in Java.
* In order to use data structures and collections in Java, we need util class because it contains the definition of all data structures and collections.
* In order to work with pipes and to read data from files, we need the io library. It allows developers to use files in their Java applications.
* The nio is another library that stands for non-blocking I/O and is an alternative to java.io library. By using it, we can get the advantage of intensive use of I/O operations.
* The math library is one of the libraries used for mathematical calculation, such as the sum of BigInteger or BigDecimal.
* In order to work with networks, connections, and sockets, java.net provides all the required classes for it. The net library is mostly used for developing network applications.
* The swing and java.awt are two libraries used to create GUI (Graphical User Interface). The java.awt is available in the older version of Java.
* Sound is another library that is used for media content.

|  |  |
| --- | --- |
| **EX.NO : 01** | **IMPLEMENTATION OF DIFFERENT OPERATIONS USING STRING AND STRING BUFFER** |
| **DATE :25.01.2023** |

**LOW**

Develop a Java program to reverse words in a given string**.**

**AIM:**

To reverse words using java program.

**PROCEDURE:**

**Step 1** Start the process

**Step 2** Read the input from user.

**Step 3** Split the input string into an array of words using the split() method with a space (" ") as the delimiter.

**Step 4** Create an empty string variable to store the reversed string.

**Step 5** Iterate through the array of words in reverse order using a loop.

**Step 6** Inside the loop, append each word to the reversed string variable followed by a space.

**Step 7** Finally, print the reversed string.

**CODE:**

import java.util.regex.Pattern;

public class Example {

public static void main(String[] args) {

Scanner s=new Scanner (System.in);

String str = s.Line();

Pattern p = Pattern.compile("\s");

String[] temp = p.split(str);

String rev = "";

for (int i = 0; i < temp.length; i++) {

if (i == temp.length - 1)

rev = temp[i] + rev;

else

rev = " " + temp[i] + rev;

}

System.out.println("The reversed strev)rev);

}

}

**INPUT :**

I am Dhanush

**OUTPUT:**

Dhanush am I

**RESULT:**

Thus the java program the reverse words in a given stringis found successfully.

**MEDIUM**

Develop a Java program to compare two strings lexicographically,ignoring case differences

**AIM:**

Compare two strings lexicographically,ignoring case differences or not

**PROCEDURE:**

**Step 1**  Start the process.

**Step 2**  Read the two strings a,b from user.

**Step 3** Using equalsIgnoreCase()function check

If its equalsIgnoreCase then print a is equal to b

Else print a is not equal to b

**Step 4**  Stop the process.

**CODE:**

import java.util.\*;

public class com

{

public static void main(String args[])

{

Scanner s=new Scanner (System.in);

String a=s.nextLine();

String b=s.nextLine();

if(a.equalsIgnoreCase(b))

{

System.out.print(a+" is equal to "+b);

}

else

{

System.out.print(a+" is not equal to "+b);

}

}

}

**INPUT :**

I Am Dhanush

I am dhanush

**OUPUT:**

I Am Dhanush is equal to I am dhanush.

**RESULT:**

Thus the Java Program the two strings is checked lexicographically,ignoring case differences or not.

**HIGH**

Develop a java program to create a StringBuffer object, append characters display the capacity and length of the string buffer.

**AIM:**

To create a StringBuffer object, append characters display the capacity and length of the string buffer

**PROCEDURE:**

**Step 1** Start the process.

**Step 2** Create a StringBuffer object and store the String

**Step 3** Using append function append the String to the old String

**Step 4** Using Length and Capacity function find the length and capacity of the string

**Step 5** Print all the values

**Step 6** Stop the process.

**CODE:**

import java.util.\*;

public class buffertest

{

public static void main (String args [])

{

StringBuffer buf=new StringBuffer("DHANUSU");

buf.append(" GOPAL");

System.out.println(buf);

System.out.println("Lenth Of The StringBuffer : "+ buf.length());

System.out.print("Capacity Of The StringBuffer : "+buf.capacity());

}

}

**INPUT:**

Dhanusu gopl

**OUPUT:**

Dhanusugopl

Lenth Of The StringBuffer : 13

Capacity Of The StringBuffer : 23

**RESULT:**

Thu the Java Program the StringBuffer object is created , append characters display the capacity and length of the string buffer was implemented successfully.

**MARK ALLOCATION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PARMETER** | **MARKS ALLOCATED** | **MARKS OBTAINED** | | |
| **Low**  **(20 Marks)** | **Medium**  **(30 Marks)** | **High**  **(50 Marks)** |
| **A** | **30** |  |  |  |
| **B** | **30** |  |  |  |
| **C** | **30** |  |  |  |
| **D** | **10** |  |  |  |
| **Sub Total** | |  |  |  |
| **Total (100)** | |  | | |

**STAFF SIGNATURE**

|  |  |
| --- | --- |
| **EX NO : 02** | **DEMONSTRATE VARIOUS CLASSES AND INTERFACES OF COLLECTIONS** |
| **DATE : 06.09.2022** |

**LOW**

Develop a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' without any parameter in its constructor.

**AIM:**

Develop a program to print the area and perimeter of a triangle using constructor.

**PROCEDURE:**

**Step 1:** Start the process.

**Step 2:** Create class Triangle and using constructor declare a=3,b=4,c=5

**Step 3:** In cal function find the area,perimeter of the triangle print the values.

**Step 4:** In min class created object.

**Step 5:** Call the function.

**Step 6:** Stop the process.

**CODE:**

import java.util.\*;

class triangle{

int a,b,c;

double area,perimeter;

triangle()

{

a=3;

b=4;

c=5

}

public void cal(){

area=0.5\*a\*b;

perimeter=a+b+c;

System.out.println(perimeter);

System.out.print(area);

}

};

class test{

public static void main(String args[])

{

triangle t= new triangle();

t.cal();

}

}

**INPUT :**

3 4 5

**OUTPUT :**

12

6

**RESULT :**

Thus, the program the area and perimeter of a triangle has been executed successfully.

**MEDIUM**

A Java abstract class is a class that can't be instantiated. That means you cannot create new instances of an abstract class. It works as a base for subclasses. You should learn about Java Inheritance before attempting this challenge.

Following is an example of abstract class:

abstract class Book{

String title;

abstract void setTitle(String s);

String getTitle(){

return title;

}

}

If you try to create an instance of this class like the following line you will get an error:

Book new\_novel=new Book();

You have to create another class that extends the abstract class. Then you can create an instance of the new class. Use setTitle method is abstract too and has no body. That means you must implement the body of that method in the child class. Create a class named as MyBook.

**AIM :**

To develop a Java program for Abstract class and display the Following

**PROCEDURE:**

**Step 1 :** Start the process.

**Step 2 :** Create an Abstract class and Declare the methods only.

**Step 3 :** Create a Class extends with abstract class and implement the methods where declare in the abstract class.

**Step 4 :** In min class created object.

**Step 5 :**Call the function.

**Step 6 :** Stop the process.

**CODE:**

import java.util.\*;

abstract class book

{

String title;

abstract void gettitle(String a);

String display()

{

return title;

}

}

class mybook extends book

{

void gettitle(String a)

{

title=a;

}

}

public class test

{

public static void main(String args[])

{

Scanner o=new Scanner (System.in);

String a=o.nextLine();

mybook obj=new mybook();

obj.gettitle(a);

System.out.print("The title is: "+obj.display());

}

}

**INPUT :**

The Java Progrmming

**OUTPUT :**

The title is : The Java Programming

**RESULT :**

Thus, the program the abstract class was implemented successfully.

**HIGH:**

A Java interface can only contain method signatures andFields. The interface can be used to achieve polymorphism. In this problem, you will practice your knowledge on interfaces.

You are given an interface Advanced Arithmetic which Contains a method signature int divisor\_sum(int n). You need to write a class called My Calculator which implements the interface, divisor Sum function just takes an integer as input and return the sum of all its divisors

For example divisors of 6 are 1, 2, 3 and 6,

So divisor\_sum should return 12. The value of n will be at most 1000.

Read the partially completed code in the editor and complete it.

You just need to write the My Calculator class only. Your class shouldn’t be public

Explanation

Divisors of 6 are 1,2,3 and 6. 1+2+3+6=12.

**AIM :**

To develop a Java program to find the sum of the divisors of the number.

**PROCEDURE:**

**Step 1** : Start the process.

**Step 2 :** Read the number.

**Step 3 :** Create a interface which contains the declaration of the ArrayList and the method

divisor\_sum().

**Step 4 :** Create a class MyCalculator the implements the interface.

**Step** **5** **:** Override the method to find the sum of the divisors.

**Step 6 :** Print the result.

**Step 7 :** Stop the process.

**CODE:**

import java.util.\*;

interface AdvancedArithmetic{

ArrayList<Integer> arr = new ArrayList<Integer>();

String str = “I implemented: AdvancedArithmetic”;

public void divisor\_sum(int n);

}

class MyCalculator implements AdvancedArithmetic{

int sum = 0;

//int divisors[] = new int[size];

public void divisor\_sum(int n){

for(int i=1;i<=n;i++){

if(n % I == 0){

arr.add(i);

}

}

}

public int divisorSum(){

for(int i=0;i<arr.size();i++){

sum += arr.get(i);

}

return sum;

}

}

public class Calculator{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

MyCalculator obj = new MyCalculator();

obj.divisor\_sum(n);

System.out.println(obj.str);

System.out.print(obj.divisorSum());

}

}

**INPUT:**

6

**OUTPUT:**

12

**RESULT:**

Thus, the program to print the sum of the divisors of the number has been executed successfully.

**MARK ALLOCATION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PARMETER** | **MARKS ALLOCATED** | **MARKS OBTAINED** | | |
| **Low**  **(20 Marks)** | **Medium**  **(30 Marks)** | **High**  **(50 Marks)** |
| **A** | **30** |  |  |  |
| **B** | **30** |  |  |  |
| **C** | **30** |  |  |  |
| **D** | **10** |  |  |  |
| **Sub Total** | |  |  |  |
| **Total (100)** | |  | | |

**STAFF SIGNATURE**

|  |  |
| --- | --- |
| **EX NO : 03** | **IMPLEMENTATION OF DIFFERENT APPLICATIONS USING PACKAGES AND TO CHECK ABNORMAL CONDITIONS USING EXCEPTION HANDLING** |
| **DATE : 14.02.2023** |

**LOW**

Develop the code to implement the Arithmetic Exception using Throws and Finally

**AIM:**

To develop a Java program to implement the Unchecked Exception with multiply try and catch block.

**PROCEDURE:**

**Step 1 :** Start the process.

**Step 2 :** Read the input from user.

**Step 3 :** Use a try-catch block to handle any exceptions that may occur during the conversion or parsing process.

Within the try block:

- Convert the input string to a numeric type (e.g., int, float, etc.).

- If the number is greater than 0:

- Display "The number is positive."

- Else, if the number is less than 0:

- Display "The number is negative."

- Else:

- Display "The number is zero."

b. Catch a NumberFormatException:

- Display "Invalid input. Please enter a valid number."

.

**Step 4:** Stop the process.

**CODE:**

import java.util.\*;

import java.lang.Exception;

class myException extends Exception

{

myException(String message)

{

super(message);

}

}

class Test

{

public static void main(String args[])

{

Scanner o=new Scanner(System.in);

int a=o.nextInt();

try

{

if(a>0)

{

throw new myException ("the number is positive");

}

else

{

throw new myException("the number is negative");

}

}

catch(myException e)

{

System.out.println("caught my exception");

System.out.println(e.getMessage());

}

finally

{

System.out.println("the given num is checked");

}

}

}

**INPUT :**

5

**OUTPUT :**

caught my exception

the number is positive

the given num is checked

**RESULT :**

Thus, the program to handle the Unchecked Exception has been executed successfully.

**MEDIUM:**

You will be given two integers and as input, you have to compute If and are not bit signed integers or if is zero, exception will occur and you have to report it. Read sample Input /Output to know what to report in case of exceptions .Sample Input /Output:

Input:

10

3

Output:

3

Input:

10

Hello

Output:

Java.util.InputMismatchException

Input:

10

0

Output:

Java.lang.ArithmeticException: /by zero

Check all the test cases.

**AIM :**

To develop a Java program to handle all type of Exceptions and display what type of exceptions is occurred.

**PROCEDURE:**

**Step 1 :** Start the process.

**Step 2 :** Inside the try block read the numbers and perform division operation.

If any error is happened, then throw and appropriate error to the catch block.

**Step 3 :** If no exception is occurred, then print the result.

**Step 4 :** Stop the process.

**CODE:**

import java.util.\*;

import java.lang.Exception;

public class Exception\_All{

Public static void main(String args[]){

Scanner sc = new Scanner(System.in);

try{

int num1 = sc.nextInt();

int num2 = sc.nextInt();

int ans = num1/num2;

System.out.println(ans);

}

catch(Exception e){

System.out.println(e);

}

}

}

**INPUT:**

450

0

**OUTPUT:**

Java.lang.ArithmeticException: /by zero

**RESULT:**

Thus, the program to handle all type of exceptions occurred in the program has been executed successfully.

**HIGH:**

Write a Java program to implement method overriding with various packages.

Use private/public methods

**AIM :**

To develop a Java program to implement method overriding with various packages.

**PROCEDURE:**

**Step 1 :** Start the process.

**Step 2 :** Create a package called LibraryExercise and define the class\_A class\_B and class\_C.

**Step 3 :** Define four methods of public and private access specifiers to perform some task.

**Step 4 :** Override all the methods in the class to print the string.

**Step 5 :** Define a class main method that create objects of the classes.

**Step 6 :** Print content of the each method by calling the method using object.

**Step 7 :** Stop the process.

**CODE:**

package LabExercise;

public class Class\_A{

public void methodLab1(){

System.out.println(“Method in Class\_A is executed”);

}

public void methodLab2(){

System.out.println(“Second Method in Class\_A is executed”);

}

}

package LabExercise;

public class Class\_B{

public void methodLab3(){

System.out.println(“Method in Class\_B is executed”);

}

}

package LabExercise;

public class Class\_C{

public void methodLab4(){

System.out.println(“Method in Class\_C is executed”);

methodLab5();

}

private void methodLab5(){

System.out.println(“Private Method in Class\_C is executed”);

}

}

import java.lang.Exception;

import LabExercise.Class\_A;

import LabExercise.Class\_B;

import LabExercise.Class\_C;

class Child\_1 extends Class\_A{

public void methodLab1(){

System.out.println(“Overrided method in Child\_1”);

super.methodLab1();

}

public void methodLab2(){

System.out.println(“Overrided method 2 in Child\_1”);

super.methodLab2();

}

}

class Child\_2 extends Class\_B{

public void methodLab3(){

System.out.println(“Overrided method in Child\_2”);

super.methodLab3();

}

}

class Child\_3 extends Class\_C{

public void methodLab4(){

System.out.println(“Overrided private method in Child\_3”);

super.methodLab4();

}

}

public class Parent\_Class{

public static void main(String args[]){

try{

Child\_1 obj1 = new Child\_1();

Child\_2 obj2 = new Child\_2();

Child\_3 obj3 = new Child\_3();

obj1.methodLab1();

obj1.methodLab2();

obj2.methodLab3();

obj3.methodLab4();

}

catch(Exception e){

System.out.println(e);

}

}

}

**OUTPUT :**

Overrided method in Child\_1.

Method in class\_A is executed.

Overrided method 2 in Child\_1.

Second method in Class\_A is executed.

Overrided method in Child\_2.

Method in Class\_B is executed.

Overrided private method in Child\_3.

Method in Class\_C is executed.

Private method in Class\_B is executed

**RESULT :**

Thus, the program has been executed successfully and the outputs are verified.

**MARK ALLOCATION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PARMETER** | **MARKS ALLOCATED** | **MARKS OBTAINED** | | |
| **Low**  **(20 Marks)** | **Medium**  **(30 Marks)** | **High**  **(50 Marks)** |
| **A** | **30** |  |  |  |
| **B** | **30** |  |  |  |
| **C** | **30** |  |  |  |
| **D** | **10** |  |  |  |
| **Sub Total** | |  |  |  |
| **Total (100)** | |  | | |

**STAFF SIGNATURE**

|  |  |
| --- | --- |
| **EX NO : 04** | **IMPLEMENTATION OF MULTI-TASKING CONCEPTS USING THREADS** |
| **DATE : 01.03.2023** |

**LOW:**

Create a thread and explore built-in methods available for threads.

**AIM :**

To develop a Java program to create a bulid-in methods.

**PROCEDURE:**

**Step 1 :**  Start the process.

**Step 2 :** Create a class base which extends thread class.

**Step 3 :** Implement the method of run in the base class

**Step 4 :** use the built in method called sleep.

**Step 5 :** start the thread using start method..

**Step 6 :** Stop the process.

**CODE:**

import java.util.\*;

public class Dhanush1 implements Runnable

{

@Override

public void run()

{

}

public static void main(String args[])

{

Thread a=new Thread();

a.start();

try

{

a.sleep(1000);

}

catch(InterruptedException e)

{

e.printStackTrace();

}

a.setPriority(1);

int b=a.getPriority();

System.out.println(b);

System.out.println("THREAD RUNNING");

}

}

**OUTPUT :**

5

THREAD RUNNING

**RESULT :**

Thus, the program to implement built int method has been executed successfully.

**MEDIUM:**

Develop a code to stop the thread using interrupt () method.

**AIM :**

To develop a Java program to stop the thread.

**PROCEDURE:**

**Step 1 :** Start the process.

**Step 2 :** Create class base which extends the thread class.

**Step 3 :** Implements the run method in the extended class.

**Step 4 :** Start the thread using start method in base.

**Step 5 :** Then,call the built in method interrupt.

**Step 6 :** Stop the process.

**CODE:**

import java.util.\*;

class base extends Thread

{

public void run()

{

try{

for(int i=0;i<3;i++)

{

System.out.println(“Hello”);

Thread.sleep(1000);

}

}

catch(Exception e)

{

System.out.println€;

}

}

}

class thread

{

public static void main(String args[]) throws Exception

{

base s=new base();

s.start();

s.interrupt();

System.out.println(“world”);

}

}

**OUTPUT :**

World

Hello

Interrupt has occurred.

**RESULT :**

Thus, the program to stop the thread has been executed successfully.

**HIGH:**

Implement multiple thread using Java Create Thread A for Arithmetic operations (add,sub and mul) Create Thread B for Trigonometric operations (sin,cos and tan).

**AIM:**

To Develop a Java program for implementing multiple thread.

**PROCEDURE:**

**Step 1 :** Start the process.

**Step 2 :** Create a package called LibraryExercise and define the class\_A class\_B and class\_C.

**Step 3 :** Define four methods of public and private access specifiers to perform some task.

**Step 4 :** Override all the methods in the class to print the string.

**Step 5 :** Define a class main method that create objects of the classes.

**Step 6 :** Print content of the each method by calling the method using object.

**Step 7 :** Stop the process.

**CODE:**

import java.util.\*;

class a extends Thread

{

int a,b;

a(int x,int y)

{

a=x;

b=y;

}

public void run()

{

System.out.println("add "+(a+b));

System.out.println("sub "+(a-b));

System.out.println("mul "+(a\*b));

}

}

class b extends Thread

{

int a;

b(int x)

{

a=x;

}

public void run()

{

System.out.println("sin "+a+" "+Math.sin(a));

System.out.println("cos "+a+" "+Math.cos(a));

System.out.println("tan "+a+" "+Math.tan(a));

}

}

class dhanush

{

public static void main(String args[])

{

Scanner s=new Scanner(System.in);

int a=s.nextInt();

int b=s.nextInt();

int c=s.nextInt();

a x=new a(a,b);

b y=new b(c);

x.start();

y.start();

}

}

**INPUT :**

2 3 4

**OUTPUT :**

add 5

sub -1

mul 6

sin 4 -0.7568024953079282

cos 4 -0.6536436208636119

tan 4 1.1578212823495775

**RESULT :**

Thus, the program to implement multiple thread has been implemented successfully.

**MARK ALLOCATION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PARMETER** | **MARKS ALLOCATED** | **MARKS OBTAINED** | | |
| **Low**  **(20 Marks)** | **Medium**  **(30 Marks)** | **High**  **(50 Marks)** |
| **A** | **30** |  |  |  |
| **B** | **30** |  |  |  |
| **C** | **30** |  |  |  |
| **D** | **10** |  |  |  |
| **Sub Total** | |  |  |  |
| **Total (100)** | |  | | |

**STAFF SIGNATURE**

|  |  |
| --- | --- |
| **EX.NO:05** | **IMPLEMENTATION OF ACCESSING REMOTE DATA USING RMI.** |
| **DATE:15.03.2023** |

**HIGH**

**AIM:**

To implement of Accessing Remote Data Using RMI.

**PROCEDURE:**

**Step1:** Start

**Step2:**Create a Interface.

**Step3:**Implement the interface.

**Step4:**Create a Client class.

**Step5:**Create a Server class.

**Step6:** Stop

**CODE:**

import java.rmi.\*;

public interface Adder extends Remote{

public int add(int x,int y)throws RemoteException;

}

import java.rmi.\*;

import java.rmi.server.\*;

public class AdderRemote extends UnicastRemoteObject impleme nts Adder{

AdderRemote()throws RemoteException{

super();

}

public int add(int x,int y){return x+y;}

}

import java.rmi.\*;

import java.rmi.registry.\*;

public class MyServer{

public static void main(String args[]){

try{

Adder stub=new AdderRemote();

Naming.rebind("rmi://localhost:5000/sonoo",stub);

}catch(Exception e){System.out.println(e);}

}

}

import java.rmi.\*;

public class MyClient{

public static void main(String args[]){

try{

Adder stub=(Adder)Naming.lookup("rmi://localhost:5000/sonoo");

System.out.println(stub.add(34,4));

}catch(Exception e){}

}

}

**OUTPUT:**

38

**RESULT:**

Thus the program to implement RMI has been executed and output is verified.

**MARK ALLOCATION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PARMETER** | **MARKS ALLOCATED** | **MARKS OBTAINED** | | |
| **Low**  **(20 Marks)** | **Medium**  **(30 Marks)** | **High**  **(50 Marks)** |
| **A** | **30** |  |  |  |
| **B** | **30** |  |  |  |
| **C** | **30** |  |  |  |
| **D** | **10** |  |  |  |
| **Sub Total** | |  |  |  |
| **Total (100)** | |  | | |

**STAFF SIGNATURE**

|  |  |
| --- | --- |
| **EX.NO:6** | **IMPLEMENTATION OF INNUMERABLE TASKS USING REGEX AND STREAMS** |
| **DATE:11.04.2023** |

**HIGH:**

**AIM:**

To check if the given string is a number using regex and streams.

**PROCEDURE:**

**Step 1:** Start

**Step 2:** Create a collection list.

**Step 3:** By using streams check the numbers in the string.

**Step 4:** If numbers are found in the string print the number

**Step 5:** Stop

**CODE:**

import java.util.regex.\*;

import java.util.Arrays;

import java.util.List;

public class JavaRegEx1

{

public static void main(String[] args) {

List<String> numbers = Arrays.asList("1", "20", "A1", "333", "A2A211");

for (String number : numbers) {

if (number.matches("\\d+"))

{

System.out.println(number); // 1, 20, 333

}

}

numbers.stream()

.filter(x ->x.matches("\\d+"))

.forEach(System.out::println);

}

}

**OUTPUT:**

1

20

333

**RESULT:**

Thus the program to implement the Regex and streams has been executed and the output is verified.

**MARK ALLOCATION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PARMETER** | **MARKS ALLOCATED** | **MARKS OBTAINED** | | |
| **Low**  **(20 Marks)** | **Medium**  **(30 Marks)** | **High**  **(50 Marks)** |
| **A** | **30** |  |  |  |
| **B** | **30** |  |  |  |
| **C** | **30** |  |  |  |
| **D** | **10** |  |  |  |
| **Sub Total** | |  |  |  |
| **Total (100)** | |  | | |

**STAFF SIGNATURE**

|  |  |
| --- | --- |
| **EX.NO:7** | **IMPLEMENTATION OF SERVER PROGRAMMING USING SERVLETS.** |
| **DATE:19.04.2023** |

**HIGH:**

**AIM:**

To implement server program using Servlets.

**PROCEDURE:**

**Step 1:**Start

**Step 2:**Implement the servlet interface.

**Step 3:**By using the function service initiate the server.

**Step 4:**By using the xml code implement the server program.

**Step 5:**Stop .

**CODE:**

import java.io.\*;

import javax.servlet.\*;

public class First implements Servlet

{

ServletConfigconfig=null;

public void init(ServletConfigconfig){

this.config=config;

System.out.println("Servlet is initialized");

}

public void service(ServletRequestreq,ServletResponse res) throws IOException,ServletException

{

res.setContentType("text/html");

PrintWriter out=res.getWriter();

out.print("<html><body>");

out.print("<b>Hello this is an example of simple servlet</b>");

out.print("</body></html>");

}

public void destroy(){System.out.println("Servlet is destroyed");

}

public ServletConfiggetServletConfig()

{return config;}

public String getServletInfo()

{return "copyright 2007-1010";}

}

web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app >

<servlet>

<servlet-name>s1</servlet-name>

<servlet-class>First</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>s1</servlet-name>

<url-pattern>/hello</url-pattern>

</servlet-mapping>

<display-name>Simple</display-name>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

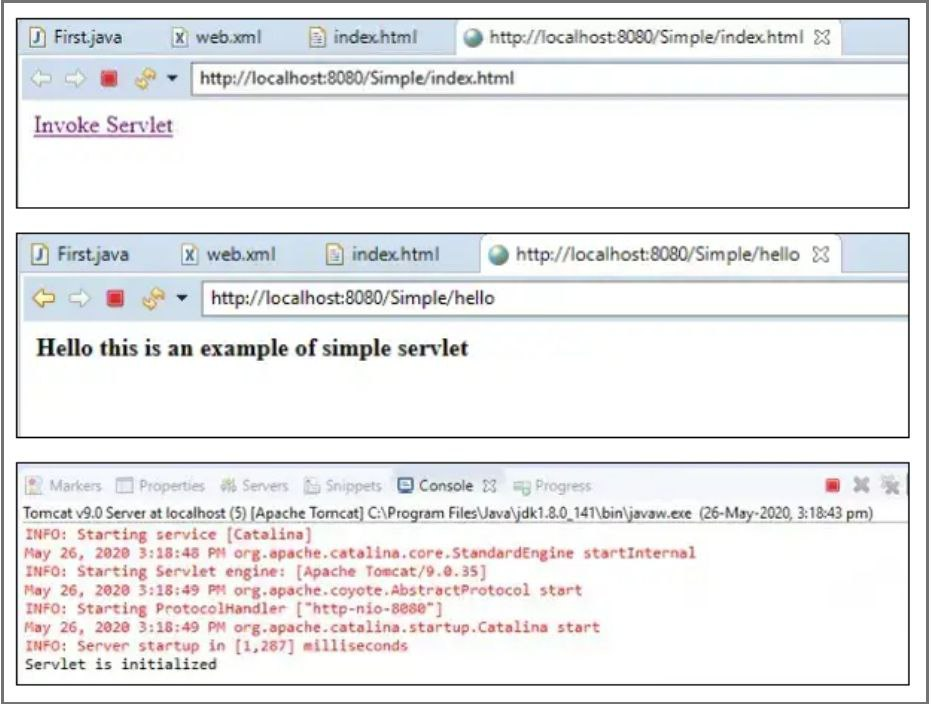
</welcome-file-list>

</web-app>

index.html

<a href="hello">Invoke Servlet</a>

**OUTPUT:**



**RESULT:**

Thus the java program to implement servlet has been executed and the output is verified.

**MARK ALLOCATION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PARMETER** | **MARKS ALLOCATED** | **MARKS OBTAINED** | | |
| **Low**  **(20 Marks)** | **Medium**  **(30 Marks)** | **High**  **(50 Marks)** |
| **A** | **30** |  |  |  |
| **B** | **30** |  |  |  |
| **C** | **30** |  |  |  |
| **D** | **10** |  |  |  |
| **Sub Total** | |  |  |  |
| **Total (100)** | |  | | |

**STAFF SIGNATURE**

|  |  |
| --- | --- |
| **EX.NO:8** | **DEMONSTRATE THE GRAPHICS APPLICATIONS USING JAVAFX** |
| **DATE:10.05.2023** |

**HIGH:**

**AIM:**

To Demonstratethe Graphics applications using JavaFx.

**PROCEDURE:**

**Step 1:** Start

**Step 2:** Import the appropriate packages

**Step 3:** By using button functions create the calculator buttons

**Step 4:** Implement the switch cases for addition, subtraction, multiplication, division.

**Step 5:** Stop .

**CODE:**

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.scene.control.Button;

import javafx.scene.control.TextField;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.ColumnConstraints;

import javafx.scene.layout.Priority;

import javafx.stage.Stage;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

public class JavaFXCalculator extends Application

{

private TextFieldtfDisplay; // display textfield

private Button[] btns; // 16 buttons

private String[] btnLabels = { // Labels of 16 buttons

"7", "8", "9", "+",

"4", "5", "6", "-",

"1", "2", "3", "x",

"C", "0", "=", "/"

};

// For computation

private int result = 0; // Result of computation

private String inStr = "0"; // Input number as String

// Previous operator: ' '(nothing), '+', '-', '\*', '/', '='

private char lastOperator = ' ';

// Event handler for all the 16 Buttons

EventHandler handler = evt -> {

String currentBtnLabel = ((Button)evt.getSource()).getText();

switch (currentBtnLabel) {

// Number buttons

case "0": case "1": case "2": case "3": case "4":

case "5": case "6": case "7": case "8": case "9":

if (inStr.equals("0")) {

inStr = currentBtnLabel; // no leading zero

} else {

inStr += currentBtnLabel; // append input digit

}

tfDisplay.setText(inStr);

// Clear buffer if last operator is '='

if (lastOperator == '=') {

result = 0;

lastOperator = ' ';

}

break;

// Operator buttons: '+', '-', 'x', '/' and '='

case "+":

compute();

lastOperator = '+';

break;

case "-":

compute();

lastOperator = '-';

break;

case "x":

compute();

lastOperator = '\*';

break;

case "/":

compute();

lastOperator = '/';

break;

case "=":

compute();

lastOperator = '=';

break;

// Clear button

case "C":

result = 0;

inStr = "0";

lastOperator = ' ';

tfDisplay.setText("0");

break;

}

};

// User pushes '+', '-', '\*', '/' or '=' button.

// Perform computation on the previous result and the current input number,

// based on the previous operator.

private void compute() {

intinNum = Integer.parseInt(inStr);

inStr = "0";

if (lastOperator == ' ') {

result = inNum;

} else if (lastOperator == '+') {

result += inNum;

} else if (lastOperator == '-') {

result -= inNum;

} else if (lastOperator == '\*') {

result \*= inNum;

} else if (lastOperator == '/') {

result /= inNum;

} else if (lastOperator == '=') {

// Keep the result for the next operation

}

tfDisplay.setText(result + "");

}

// Setup the UI

@Override

public void start(Stage primaryStage) {

// Setup the Display TextField

tfDisplay = new TextField("0");

tfDisplay.setEditable(false);

tfDisplay.setAlignment(Pos.CENTER\_RIGHT);

// Setup a GridPane for 4x4 Buttons

intnumCols = 4;

intnumRows = 4;

GridPanepaneButton = new GridPane();

paneButton.setPadding(new Insets(15, 0, 15, 0)); // top, right, bottom, left

paneButton.setVgap(5); // Vertical gap between nodes

paneButton.setHgap(5); // Horizontal gap between nodes

// Setup 4 columns of equal width, fill parent

ColumnConstraints[] columns = new ColumnConstraints[numCols];

for (inti = 0; i<numCols; ++i) {

columns[i] = new ColumnConstraints();

columns[i].setHgrow(Priority.ALWAYS) ; // Allow column to grow

columns[i].setFillWidth(true); // Ask nodes to fill space for column

paneButton.getColumnConstraints().add(columns[i]);

}

// Setup 16 Buttons and add to GridPane; and event handler

btns = new Button[16];

for (inti = 0; i<btns.length; ++i) {

btns[i] = new Button(btnLabels[i]);

btns[i].setOnAction(handler); // Register event handler

btns[i].setMaxSize(Double.MAX\_VALUE, Double.MAX\_VALUE); // full-width

paneButton.add(btns[i], i % numCols, i / numCols); // control, col, row

}

// Setup up the scene graph rooted at a BorderPane (of 5 zones)

BorderPane root = new BorderPane();

root.setPadding(new Insets(15, 15, 15, 15)); // top, right, bottom, left

root.setTop(tfDisplay); // Top zone contains the TextField

root.setCenter(paneButton); // Center zone contains the GridPane of Buttons

// Set up scene and stage

primaryStage.setScene(new Scene(root, 300, 300));

primaryStage.setTitle("JavaFX Calculator");

primaryStage.show();

}

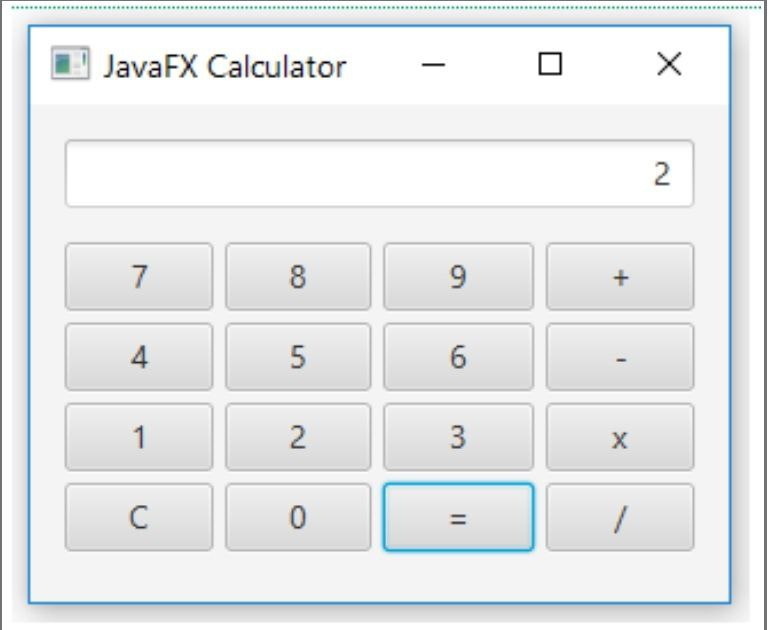
public static void main(String[] args) {

launch(args);

}

}

**OUTPUT:**



**RESULT:**

Thus java program to Demonstrate the graphics using JavaFx has been executed and the output is verified.

**MARK ALLOCATION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PARMETER** | **MARKS ALLOCATED** | **MARKS OBTAINED** | | |
| **Low**  **(20 Marks)** | **Medium**  **(30 Marks)** | **High**  **(50 Marks)** |
| **A** | **30** |  |  |  |
| **B** | **30** |  |  |  |
| **C** | **30** |  |  |  |
| **D** | **10** |  |  |  |
| **Sub Total** | |  |  |  |
| **Total (100)** | |  | | |

**STAFF SIGNATURE**