**Department of Computer Engineering**

**Academic Term : July-Nov 2018**

**LAB MANUAL**

**Class: *S.E. Computer***

**Subject Code: *CSL304***

**Subject Name: *OOPM Lab***

|  |  |
| --- | --- |
| **Title** | **Program on branching, looping.** |
| Aim | Write a program to display the number pattern. For n = 3     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  | **1** |  |  | |  | **2** |  | **2** |  | | **3** |  | **3** |  | **3** | |
| Theory/ Algorithm | 1. **For Statement:** 2. **While Statement: 3. Do-While Statement:**       **4. Labelled Break Statement: 5. Labelled Continue Statement:**  Logic of displayPattern() function  for row=1 to **\_\_\_**  repeat the steps below (every row)   1. for col=1 to \_\_\_\_\_\_ (numofspaces of currentrow) 2. Print a space 3. for col=1 to \_\_\_\_\_ (numofnumbers of currentrow) 4. Print colnumber |
| Test Cases  (Important) |  |
| Results and Conclusion |  |
| Post Lab | Draw JVM component diagram and list down the function of each component. |
| |  |  | | --- | --- | | **Title** | **To solve the computational problem using basic control structures.** | | Aim | Write a program to calculates LCM/GCD of two numbers. | | Theory/ Algorithm | input variable : n1, n2 output variable: lcm, gcd | | Class  Diagram |  | | Test Cases  (Important) |  | | Results and Conclusion |  | | Post Lab | Explain three different ways to read input data in Java. | | |
| **Title** | **Program on various ways to accept data through keyboard .** |
| Aim | Write a program that takes three numbers as input arguments through command line.  It finds the maximum of three. |
| Theory/ Algorithm | **Conditional Operator:**   1. Command that you will use to run **Max.java** numbers program with arguments:   **$ javac Max.java**  **$ java Max 5 8 15**  **15**   1. **Command Line Arguments is Java**: Command Line Arguments in Java are the parameters passed to main function through command line, when we run the program. (i.e when we invoke java interpreter using $java command). These arguments are stored in the String array args, that we use in main function definition 2. public static void main(**String args[]**) 3. If three parameters are passed then they are stored as the args[0], args[1] and args[2].   **args.length** would give the number parameters passed.  Command to convert String to int primitive datatype:  int n = **Integer.parseInt(s1)**; where s1 is String object  Command to convert String to char primitive datatype:  char ch = s1.charAt(0); where s1 is String object |
| Class  Diagram |  |
| Test Cases  (Important) |  |
| Conclusion |  |
| Post Lab | Explain how java is architecture neutral? |

|  |  |
| --- | --- |
| **Title** | **Program to create class with members and methods, accept and display details for single object. (Use Constructor and Constructor Overloading)** |
| Aim | Write a class **Employee**  for a company and display details. Modify it with static variable. |
| Theory/ Algorithm | **Define:**  **Class:**  **Object:** |
| Class Diagram |  |
| Test Cases |  |
| Results and Conclusion |  |
| Post Lab | Explain the concept of constructor and destructor in Java. |
| |  |  | | --- | --- | | **Title** | **Recursive Function** | | Aim | Write a Program to calculate x ^n using recursion and static function.. | | Theory | Recursion- It is the technique where the function calls itself.  There should be some exit condition to exit the recursion and return to caller.  Recursive functions are easy to write but they take more time and space to  execute. | | Class  Diagram  And  Logic of the function |  | | Test Cases |  | | Results and Conclusion |  | | Post Lab | List the advantages and disadvantages of Recursion. | | |
| **Title** | **Program on method overloading** |
| Aim | Write a class **Shape** with overloaded functions that calculate area of different shapes. |
| Theory | Function Overloading: |
| Class  Diagram |  |
| Test Cases |  |
| Results and Conclusion |  |
| Post Lab | Explain Compile Time Polymorphism in Java. |
| **Title** | **Constructors and Passing Object as a parameter to function.** |
| Aim | Write a Program to add two complex numbers. |
| Theory/ Algorithm | 1. Three types of Constructors: 2. 1. Default constructor: Complex( ){ real=0.0; imag=0.0;} 3. 2. Parameterized: Complex(float real, float imag) 4. { this.real=real; this.imag=imag;} 5. 3. Copy constructor: Complex(Complex c) // constructor definition 6. { this.real = c.real; this.imag=c.imag; }   Complex c1 = new Complex(4.2,5.0); // parameterized call  Complex c2 = new Complex(c1); // copy constr call  The Objects can be passed as a parameter in java using call by reference.  The primitive data types are passed by value.  The non-primitive data types such as an array, string and objects are passed as a reference. That means once an object is created on heap, its reference is passed from caller function to callee. Even the callee can return the reference of an object created in it to the caller. |
| Class  Diagram | Output : Complex object c3. |
| Test Cases |  |
| Results and Conclusion |  |
| Post Lab | Explain the call by value and call by reference in Java |
| **Title** | **Program on Array .** |
| Aim | Write a program to print the employee of an organization in ascending order of their salary. |
| Theory/ Algorithm | An array is a list of elements of same types referred by common name. Each element is accessed through its index. We can create an array of  primitive data types such as***int, char, float, double*** and  non-primitive data types such as ***objects, interfaces, other array, Strings****.*  There are three operations we need to do with an array to use them.  Declaration of an array: *int[] a;*  Construction: *a = new int[5];*  Initialization: *for(int i =0; i<****a.length****; i++)*  *a[i]= Integer.parseInt(br.readLine());*  Each array has a property called ***length*** that gives the size of an array.  Declare array of Emp Objects: *Emp[] ea;*  Construction in constructor ***ea=new Emp[5]****; // array dimension*  Initialization getData fun( ) *for(int i=0; i<ea.length; i++)*  *{ …… read name and sal……..*  *// constructor call ea[i]=new* ***Emp( name, sal);*** *}* |
| Class  Diagram |  |
| Test Cases  (Important) |  |
| Results and Conclusion |  |
| Post Lab | Show Stack and Heap Memory Allocation for this program. |
| **Title** | **Program on 2D Array.** |
| Aim | Write a program that defines a **Matrix** class with the functions such as readMatrix, displayMatrix, printTranspose, isSymmetric and Multiplication. Then using main function invoke these functions. |
| Theory/ Algorithm | **THEORY:** In Java, *multidimensional arrays* are actually arrays of arrays.  Declaration: *int[][] a;*  Construction of even 2-d array: *a = new int[r][c];*  Initialization: *for(int i=0;i<r;i++)*  *for(int j=0;j<c;j++)*  *a[i][j]=Integer.parseInt(br.readLine());*  Logic:   1. Define a class Matrix. 2. Member variable : two-d array. 3. Member functions : constructors, displayMatrix, printTranspose, isSymmetric and multiplyMat. 4. Create matrix m1 and m2. Then call the functions. |
| Class  Diagram |  |
| Test Cases  (Important) |  |
| Results and Conclusion |  |
| Post Lab | Show the stack and heap memory allocation for this program. |

|  |  |
| --- | --- |
| **Title** | **Program on String.** |
| Aim | Write a program to count no of letters, digits, spaces, special characters in a string. |
| Theory/ Algorithm | In Java, *String* isan in-built class.  Declaration: *String s1;*  Construction String object: *s1 = new String(“Hello”);*  Functions of String class*: length, toUpperCase(), toLowerCase(),*  *substring(),concat(),replace(), chatAt(),getChars()*  *toCharArray(), equals(), indexOf, lastIndexOf()*  *compareTo(), valueOf(), toString()….*  String Literal *String s2;*  s*2=”Hello”;*  *note: s1==s2 is false*  *s1.equals(s2) is true* |
| String  Class  In-built  Functions  Contd.. next expt. |  |
| Test Cases  (Important) |  |
| Results and Conclusion |  |
| Post Lab | String class and its member functions with the meaning. (refer java docs) |

|  |  |
| --- | --- |
| **Title** | **Program on StringBuffer.** |
| Aim | Write a program that compares two Strings using String and StringBuffer class |
| Theory | **String Class Constructors:**  **StringBuffer Class Constructors:** |
| Class  Diagram/  String class  In-built functions |  |
| Test Cases  (Important) |  |
| Results and Conclusion |  |
| Post Lab | What is the difference between String and StringBuffer Class. |
| **Title** | **Program on Vector** |
| Aim | Simulate the shopping cart for the operations add Item, remove Item, Pay the bill and exit. It allows the customer to add/remove the product item in the cart. Finally display the bill for the items selected by the customer. |
| Theory/ Algorithm | Util Package (List of classes) |
| Logic/  Class Diagram |  |
| Results and Conclusion |  |
| Post Lab | List down important functions of all collection classes. |

|  |  |
| --- | --- |
| **Title** | **Program on single inheritance (Use super keyword)** |
| Aim | Write a program to implement single inheritance. |
| Theory/ Algorithm | Inheritance is an Object Oriented Programming feature that allows the programmers to create the subclass from a superclass by inheriting the some properties of superclass.  Subclass can override some superclass functionality and even add its own new functionality. Their needs to be **is-a** kind of relationship between classes.  Reasons of using inheritance:   1. We want to reuse existing code at the same time want to add extra functionality to it. 2. We want to refine the existing code without modifying existing application. 3. Generalization-Specialization relation is needed like Car Maruti. |
| Class  Diagram | 1. **Clock DigitalClock** is an example of single level inheritance |
| Test Cases |  |
| Results and Conclusion |  |
| Post Lab | What is the meaning of this and super keyword? |

|  |  |
| --- | --- |
| **Title** | **Program on multilevel inheritance, constructor and method overriding.** |
| Aim | Implement multilevel hierarchy of classes person-employee-manager. |
| Theory/ Algorithm | Multilevel Inheritance: |
| Class  Diagram | **Person Employee Manager** |
| Test Cases  (Important) |  |
| Results and Conclusion |  |
| Post Lab | What are different access specifiers? Explain them. |

|  |  |  |
| --- | --- | --- |
| **Title** | **Program on abstract class and hierarchical Inheritance.** | |
| Aim | Implement hierarchical inheritance for the classes SavingsAccount and CurrentAccount. | |
| Theory/ Algorithm | Hirerarchical Inheritance: | |
| Class  Diagram | **Account SavingsAccount and CurrentAccount** is an example of hierarchical inheritance. | |
| Test Cases  (Important) |  | |
| Results and Conclusion |  | |
| Post Lab | What is an abstract class and abstract method? | |
| **Title** | **Program on interface demonstrating concept of multiple inheritance** |
| Aim | Write a program that implements the stack interface. |
| Theory/ Algorithm | In Java, Multiple Inheritance is not present. A class can inherit from only one class but it can implement multiple interfaces.  Interfaces are syntactically similar to classes, but they lack instance variables.  Interface may contain constants and methods declaration.  Once Interface is defined, any number of classes can implement that interface.  To implement the interface, the other class need not be the subclass of interface.  The other class will have to override all the methods of an interface.  However each class is free to determine the details of implementation. |
| Class  Diagram |  |
| Results and Conclusion |  |
| Post Lab | What is the difference between an interface and an abstract class. |

|  |  |
| --- | --- |
| **Title** | **Write a program that demonstrates dynamic method dispatch.** |
| Aim |  |
| Theory/ Algorithm |  |
| Class  Diagram |  |
| Results and Conclusion |  |
| Post Lab | **Explain Run Time Polymorphism.** |
| **Title** | **Program on creating user defined package** |
| Aim | Write a program that creates a "**Student**" class and "S**taff**" class in a "**college**" package.the classes are defined in two different files. create a class "**Coengg**" in a package "**institute**". In "**Coengg**" class create the objects of "**Student**" and "S**taff**" classes and invoke the methods to display details. |
| Theory/ Algorithm | **Package:** |
| Class  Diagram |  |
| Results and Conclusion |  |
| Post Lab | List in-built Packages. |
| **Title** | **Program to demonstrate try, catch, throw, throws and finally.** |
| Aim | Write a program that handles the in-built exceptions using try-catch-finally methods. |
| Theory/ Algorithm | **Exception Handling:**  **Try:**  **Catch:**  **Finally:** |
| Class  Diagram |  |
| Results and Conclusion |  |
| Post Lab | Draw the Exception Classes hierarchy with some frequently used exceptions. |

|  |  |
| --- | --- |
| **Title** | **To implement user defined Exceptions in Java** |
| Aim | Write a program to handle the user-defined exception. A mark list containing register number and marks for a subject is given. If the marks are less than zero, user defined exception; ***IllegalMarksEnteredException*** is thrown out and handled with the message “Illegal Marks Entered”. For all valid marks, the candidate will be declared  as “Pass” if the marks are equal to or greater than 40,  as “Fail” otherwise. |
| Theory/ Algorithm | **Exception Handling:**  Try-Catch-Finally Statements in Java  try  {  Thread t1 = new Thread( );  t1.sleep( );  }  catch(InterruptedException ie)  {  System.out.println(“Thread sleep is interrupted” + ie);  }  Finally  {  System.out.println(“Inside Finally Block: This is printed always”);  } |
| Class  Diagram |  |
| Test Cases |  |
| Results and Conclusion |  |
| Post Lab | What is the difference between throw and throws keywords? |

|  |  |
| --- | --- |
| **Title** | **Program on multithreading** |
| Aim | Write a program to implement multithreading. There are two threads:  \* Multiplication table (n\*300) [ Do this by extending Thread class]  \* Fibonacci series till m terms (m>100) [by implementing Runnable interface] |
| Theory/ Algorithm | **Thread Class Constructors**  **1.**  **2.**  **3.**  **Methods of Thread class** |
| Class  Diagram |  |
| Conclusion |  |
| Post Lab | *What is the life cycle of a thread?* |
| **Title** | **Program on concept of synchronization** |
| Aim | Implement Stack push/pop operations of stack using threads. |
| Theory/ Algorithm | **Wait and notify Methods of Thread Class.** |
| Class  Diagram |  |
| Conclusion |  |
| Post Lab | Explain Mutual Exclusion. |

|  |  |
| --- | --- |
| **Title** | **Program on Applet to demonstrate Graphics, Font and Color class.** |
| Aim | Write an applet to draw the following shapes with some greeting message that accept parameters.  Cone, Cylinder, Cube, Square inside a circle, Circle inside a square**.** |
| Theory/ Algorithm | There are two kinds of Java programs that we can write,  **Java Applications** (stand-alone or console based programs with *main()* method) and  **Java Applets**. - Java Applet is embedded inside HTML, runs inside \_\_\_\_\_\_ of web  browser after getting downloaded through Internet. It is an Internet-based Java program that is mostly Graphical user Interface (GUI) program which helps user to animate graphics, play sounds and design GUI components such as text box, button and radio button. It is delivered to web client in the form of java \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  It can also run by using Sun’s \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ utility to test an Applet.  An applet must be a subclass of the *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* class, which provides the standard interface between the applet and the browser environment.  **Graphics built in method:**  Line: void drawLine(int xBegin, int yBegin, int xEnd, int yEnd)  Rectangle: void drawRect(int x, int y, int w, int h)  Oval: void drawOval(int x, int y, int w, int h)  Filled Oval: void fillOval(int x, int y, int w, int h)  Arc: void drawArc(int x, int y, int w, int h, int startAngle, int arcAngle)  Filled arc: void fillArc(int x, int y, int w, int h, int startAngle, int arcAngle  Polygon: void drawPolygon(int xPoints[], int yPoints[], int nPoints)  Filled polygon: void fillPolygon(int xPoints[], int yPoints[], int nPoints) |
| Class  Diagram |  |
| Conclusion |  |
| Post Lab | Explain an applet’s life cycle with a diagram. |

|  |  |
| --- | --- |
| **Title** | **Program on passing parameters to applets** |
| Aim | Write an applet that accepts the user name as a parameter from the HTML and displays greeting message. |
| Theory/ Algorithm | **HTML Basics and Applet tag:** |
| Class  Diagram |  |
| Conclusion |  |
| Post lab | Differentiate Applet and Applications. |
| **Title** | **Program to create GUI application without event handling using AWT controls** |
| Aim | Write a Java program that creates a User Interface that enables the user to do registration using AWT controls. |
| Theory/ | **AWT Controls in Java** |
| Conclusion |  |
| Post lab |  |

|  |  |
| --- | --- |
| **Title** | **Program to create GUI application with event handling using AWT controls** |
| Aim | Write the Java program that handles events for the above GUI for all the controls. |
| Theory/ |  |
| Conclusion |  |
| Post lab | Explain Event Handling Concepts in Java |