

St. Thomas' College of Engineering and Technology

DEPARTMENT OF ELECTRICAL ENGINEERING

Laboratory Manual

Object Oriented Programming

Code: EE-694C

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

University Roll No: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

College Roll No: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Year: 3rd Semester: 6th

Session: 2019-20

**General Information**

|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name** | Object oriented Programming | **Semester** | 6 |
| **Course Code** | EE 694C | **Year with stream** | EE |
| **Course Credit** | 2 | **Session** | 2019-20 |
| **Faculty Instructor/s** |  | **Class hours and total class load** | 3 hours/week |
| **Technical Assistant/s** |  | **Laboratory** | (I1/I2) Lab |

|  |  |
| --- | --- |
| **Course objectives** | * To present their knowledge and understanding in the theories and principles of Java Technology. * To produce object oriented program/software by utilizing java techniques of object oriented programming. * To produce event driven program by using java window programming |

|  |  |  |
| --- | --- | --- |
| **Course Outcomes** | The students will be able to:  **CO1: Construct** class, subclass, object and constructors  **CO2: Apply** polymorphism and inheritance concept to implement method overloading and overriding also make use of interface and abstract class.  **CO3**: **Apply** java exception handling techniques to handle real life problems  **CO4**: **Develop** parallel programming using multithreading technique and package to organize code.  **CO5: Create** window programming in java using Applet  **CO6: Develop** a mini project to get idea of real life problem solving | Apply  Apply  Apply  Create  Create  Create |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CO** | **PO** | | | | | | | | | | | | **PSO** | | |
| **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| CO1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CO2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CO3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CO4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CO5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CO6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| **Recommended books** | * Java - The Complete Reference, by HerbertSchildt, 7th Edition McGraw-Hill publication * Introduction to Java Programming by Y. Daniel Liang, 9thEdition,pearson publication * Thinking in Java by Bruce Eckel |

**Grading:**

**Grading to be done as per university rule**

**List of Experiments**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Lab. Session** | **Name of Assignment** | **Page Number** | **Date of Expt.** | **Grade awarded** | **Signature** |
| 01 | Introduction to Java Programming |  |  |  |  |
| 02 | Class & Object |  |  |  |  |
| 03 | Method overloading Constructor overloading |  |  |  |  |
| 04 | Static keyword |  |  |  |  |
| 05 | Inheritance |  |  |  |  |
| 06 | Abstract class |  |  |  |  |
| 07 | Interface |  |  |  |  |
| 08 | Package |  |  |  |  |
| 09 | Exception Handling |  |  |  |  |
| 10 | Multithreading |  |  |  |  |
| 11 | String handling |  |  |  |  |
| 12 | File Handling, Wrapper Class |  |  |  |  |
| 13 | Applet |  |  |  |  |
| 14 | AWT & Event Handling |  |  |  |  |
| 15 | Micro Project |  |  |  |  |

**LAB SESSION 01**

**Title: Introduction to Java Programming**

**OBJECTIVE:**

* Understand fundamentals of object-oriented programming in Java.
* To be able to use the Java SDK, how to compile and run simple Java programs.
* User input by command line argument.

**THEORY:**

**Java Path Set:**

For setting the permanent path of Java installed directory, you need to follow these steps:

**Go to MyComputer properties -> advanced tab -> environment variables ->Go to System Variables-> double click on path -> write path of bin folder in variable value -> ok -> ok -> ok**

# Java Command Line Arguments:

The java command-line argument is an argument i.e. passed at the time of running the java program. The arguments passed from the console can be received in the java program and it can be used as an input.

**Example:**

**class** A{

**public** **static** **void** main(String args[]){

**for**(**int** i=0;i<args.length;i++)

System.out.println(args[i]);  }  }

compile by > javac A.java

run by > java A sonoo jaiswal 1 3 abc

**Output:**

Your first argument is:

sonoo

jaiswal

1

3

abc

Let us look at a simple code that will print the words ***Hello World***.

*public class MyFirstJavaProgram {*

*/\* This is my first java program.*

*\* This will print 'Hello World' as the output*

*\*/*

*public static void main(String []args) {*

*System.out.println("Hello World"); // prints Hello World*

*}*

*}*

**Problem statement:**

1. Write a program to print “Welcome to JAVA Programming”.
2. Write a program to take two integers and print their sum, difference, and product.
3. Accept an integer from user(using command line) and check whether it is Prime or not.
4. Write a program to read three sides of a triangle from the user (using command line) and calculate perimeter and area of the triangle.
5. Accept a number(n) and a string and display the string , n-times.
6. Write a program that converts a Fahrenheit degree to Celsius.

**Program:**

**Discussions:**

**Questionnaires:**

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**LAB SESSION 02**

**Title: Class & Object**

**Objective:**

* Defining classes and creating object and invoking methods of the defined classes.
* Taking user input throughJava BufferedReader class and Scanner Class.

**Theory:**

**Object in Java**

An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc.

**Class in Java**

A class is a group of objects that has common properties. It is a template or blueprint from which objects are created.

**BufferedReader and Scanner class**

**I/O BufferedReader class**

BufferReader in = new Bufferedreader(new InputStreamReader(System.in));  
System.out.print("how many quizzes do you want to compute?");  
int x = Integer.parseInt(in.readLine());

**I/O Scanner class**

**public** **static** **void** main(String args[]){

Scanner sc=**new** Scanner(System.in);

System.out.println("Enter your rollno");

**int** rollno=sc.nextInt();

System.out.println("Enter your name");

String name=sc.next();

**Problem statement:**

a) Create a Box class and add method called set Data () to initialised the data member. Add another method that will return volume of the Box. Using a Demo class to demonstrate by creating two box objects and find which box having bigger size.

b) Create a class called Bank\_Account with following data member and member function

i) Account\_Holder\_Name, Account\_Number, Account\_Type(S for savings, C for current),

balance

ii) withdraw\_money() and deposit\_money with proper prototype.

c) For initialising and displaying data write proper setData() and displayData() function.

Create a Demo class and demonstrate by creating 2 objects inside main() function.

**Program:**

**Discussions:**

**Questionnaires:**

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**LAB SESSION 03**

**TITLE: : Method Overloading**

**OBJECTIVE:**

* To write java program to demonstrate method overloading for basic problems.
* To initialize object using Constructor and creating different objects.

**Theory:**

If a class has multiple methods by same name but different parameters, it is known as **Method Overloading.**

|  |
| --- |
| **Overloaded method can be differentiated by:**   * Different number of arguments. * Different type of arguments. * Different order of arguments, even if the number and types are same.   **Note:**Overloaded method can have same or different return type. |

**Example**

class Car

{

String name ;

String model;

Car( ) //Constructor

{

name ="";

model="";

}

}

Car c = new Car() //Default constructor invoked

Car c = new Car(name); //Parameterized constructor invoked

## Constructor Overloading in Java

|  |  |  |
| --- | --- | --- |
| **Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter lists**  **NOTE: The rules for constructor overloading is same as method overloading.**  **Problem statement:**  i) Create a Box class with data members and all possible types of constructors. Using a display method demonstrate by calling each types of constructor.  ii) Create a Box class having a default constructor function Box (), two parameterised constructor Box(int h,int w,int d) and Box(int i) also a member function to calculate total surface area and return the value. Using a Demo class demonstrate the above by creating two object of the above class.  iii) Create a Box class having a constructor like, Box (Box ob). Add a method to check and return whether two Box object are equal dimension or not.  iv) Demonstrate method overloading with a suitable example.  **Program:**  **Discussions:**  **Questionnaires:**  **The assignment covers COs \_\_\_\_\_\_\_\_**   |  | | --- | | **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |   **LAB SESSION 04**  **TITLE: Static Keyword**  **OBJECTIVE:**   * **To demonstrate the utility of static method and variables in java programming.**   **Theory:**  **The static keyword in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class.**  **class Languages {**  **public static void main(String[] args) {**  **display();**  **}**    **static void display() {**  **System.out.println("Java is my favorite programming language.");**  **}**  **}**  **Output**  **Java is my favorite programming language.**  **Problem statement:**  a) Create a class *Complex* for storing the real and imaginary parts of a complex number. The class should have static members to store the number of complex numbers created and display it. Write functions (static or non-static as required) to calculate i) sum, difference and product of two complex numbers, ii) modulus and complement of a complex number Define required constructors and destructors of the class. Implement the class by a main function.  b) Create a class *Point* for storing the co-ordinates of a geometrical point. Write static function to calculate the distance between two points.  **Program:**  **Discussions:**  **Questionnaires:**   |  | | --- | | **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |   **The assignment covers COs \_\_\_\_\_\_\_\_** |
| **LAB SESSION 05**  **Title: Inheritance**  **Objective:**   * Understanding the concepts of inheritance using extends keyword. * Exploring the code reusability by the concept of inheritance in a class.   **Theory:**  Inheritance in java is a mechanism in which one object acquires properties and behaviors of parent object. The extends keyword indicates that you are making a new class that derives from an existing class. In the terminology of Java, a class that is inherited is called a super class. The new class is called a subclass.  **Usage of java super Keyword**   * super is used to refer immediate parent class instance variable. * super() is used to invoke immediate parent class constructor. * super is used to invoke immediate parent class method.   **Method Overriding**  If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in java.  **final keyword**  The final keyword in java is used to restrict the user. The java final keyword can be used in many context. Final can be a  Final variable// It will be constant.  Final method // cannot be override.  Final class// cannot be extend or inherit.  **Problem statement:**  Demonstrate method overriding with a suitable example.  **Program:**  **Discussions:**  **Questionnaires:**   |  | | --- | | **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |   **The assignment covers COs \_\_\_\_\_\_\_\_** |

**LAB SESSION 06**

**Title: Abstract class**

**Objective:**

* To write java program to achieve abstraction for class and method.
* To demonstrate the utility of “abstract” keyword.

**Theory:**

A class that is declared with abstract keyword is known as abstract class in java. It can have abstract and non-abstract methods (method with body). It needs to be extended and its method implemented. It cannot be instantiated directly.

**Example:**

abstract class A{}

A method that is declared as abstract and does not have implementation is known as abstract method

Example:

abstract void printStatus();//no body and abstract

**Problem statement:**

Create an abstract Class called TwoDShape, having a undefined method ‘area()’ and two instance member ‘dim1’ and ‘dim2’. Create a class ‘Rectangle’ using the abstract class and compute area of at least two rectangle abject.

**Program:**

**Discussions:**

**Questionnaires:**

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**LAB SESSION 07**

**Title: Interface**

**Objective:**

* To demonstrate the difference between class and interface.
* To write a program to study the advantage of class over interface.
* To understand Multiple Inheritance in java using Interface.
* To attach an interface independent of class hierarchy.

**Theory:**

An interface in java is a blueprint of a class. It has static constants and abstract methods only. The interface in java is a mechanism to achieve fully abstraction. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritances in Java.

**Example:**

File name : Animal.java \*/

interface Animal

{

public void eat();

public void travel();

}

**Problem statement:**

1. Write an interface in java. Demonstrate that all the methods of the interface must be defined in the corresponding class otherwise the class become abstract.
2. Write a program in java to demonstrate multiple inheritances is possible using Interface.
3. Show multiple inheritances are not possible using class in java.

**Program:**

**Discussions:**

**Questionnaires:**

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**LAB SESSION 08**

**Title: Package**

**Objective:**

* To write a java program to understand the steps in the creation of packages.
* To study the advantage of package in java programming.

**Theory:**

A java package is a group of similar types of classes, interfaces and sub-packages.Package in java can be categorized in two form, built-in package and user-defined package.There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Some of the commonly used built-in packages are shown in the table below :

|  |  |
| --- | --- |
| **Package Name** | **Description** |
| java.lang | Contains language support classes ( for e.g classes which defines primitive data types, math operations, etc.) . This package is automatically imported. |
| java.io | Contains classes for supporting input / output operations. |
| java.util | Contains utility classes which implement data structures like Linked List, Hash Table, Dictionary, etc and support for Date / Time operations. |
| java.applet | Contains classes for creating Applets. |
| java.awt | Contains classes for implementing the components of graphical user interface ( like buttons, menus, etc. ). |
| java.net | Contains classes for supporting networking operations. |

**Problem statement:**

Write a program in java where there are two packages “pack1” and “pack2”. Package “pack1” consist of classes ‘A’ and “B”. The second package “pack2” consist of classes “X” and “Y”. Demonstrate by writing a program to use any one class of each package from the main method.

**Program:**

**Discussions:**

**Questionnaires:**

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**LAB SESSION 09**

**Title: Exception Handling**

**Objective:**

* To write java program to implement the concept of exception handling.
* How to use well-structured try, catch, and finally blocks, you can create programs that fix exceptions.
* To write a java program to implement the concept of exception handling.
* How to use well-structured try, catch, and finally blocks, you can create programs that fix exceptions.

**Theory:**

Exception is an error event that can happen during the execution of a program and disrupts its normal flow. Java provides a robust and object oriented way to handle exception scenarios, known as **Java Exception Handling**.

In java, exception handling is done using five keywords, **try, catch, throw, throws and finally**

try {    // Protected code

}

catch (ExceptionName e1)

{    // Catch block

}

**Problem statement:**

1. Demonstrate Divided-By-Zero exception by accepting user input.
2. Demonstrate “ArrayIndexOutOfBounds” exception by accepting user input.
3. Demonstrate “NumberFormatException” exception by accepting user input.
4. Write a user-define exception which will throw when a particular situation occurs, demonstrate with an example.

**Program:**

**Discussions:**

**Questionnaires:**

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**LAB SESSION 10**

**Title: Multithreading**

**Objective:**

* To create java programs in a multithread environment and implement and use the life-cycle methods related to java thread.

**Theory:**

Multithreading in java is a process of executing multiple threads simultaneously. Thread is basically a lightweight sub-process, a smallest unit of processing. A thread can be in one of the five states. The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

* New
* Runnable
* Running
* Non-Runnable (Blocked)
* Terminated

There are two ways to create a thread:

* By extending Thread class
* By implementing Runnable interface

**Problem statement:**

1. Create two threads by using Thread class and set the priority value maximum to one and minimum to other. One thread will execute odd numbers and other even numbers in the range ‘1 to 20’.
2. Show threads execution is asynchronized by default. Explain with result analysis.

**Program:**

**Discussions:**

**Questionnaires:**

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**LAB SESSION 11**

**Title: String Handling**

**Objective:**

* To write java programs to demonstrate the use of different java string functions

**Theory:**

In java, string is basically an object that represents sequence of char values. An array of characters works same as java string.

For example:

char[] ch={'j','a','v','a','t','p','o','i','n','t'};

String s=new String(ch);

is same as:

String s="Welcome";

Java String class provides a lot of methods to perform operations on string such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

In java, string objects are immutable. Immutable simply means unmodifiable or unchangeable.

To create String object

|  |
| --- |
| There are two ways to create String object:   * By string literal * By new keyword |

1) String Literal

Java String literal is created by using double quotes. For Example:

String s="welcome";

2) By new keyword

String s=new String("Welcome");//creates two objects and one reference variable

Java String class methods

**Problem statement:**

1. Write a java programming to find number of characters, words and lines in a given paragraph.
2. Find the occurrence of a character in a given line.

**Program:**

**Discussions:**

**Questionnaires:**

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**LAB SESSION 12**

**Title: File Handling & Wrapper Class**

**Objective:**

Demonstrate simple file handling in java.

Demonstrate wrapper class w.r.t primitive data types.

**Theory1:**

In Java, FileInputStream and FileOutputStream classes are used to read and write data in file. In another words, they are used for file handling in java.

Java FileWriter and FileReader classes are used to write and read data from text files. These are character-oriented classes, used for file handling in java. Java has suggested not to use the FileInputStream and FileOutputStream classes if you have to read and write the textual information.

**import** java.io.\*;

**class** Simple{

**public** **static** **void** main(String args[]){

**try**{

   FileWriter fw=**new** FileWriter("abc.txt");

   fw.write("my name is sachin");

   fw.close();

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

  System.out.println("success");

 }

}

**Output**

success

**Theory2:**

Wrapper class in java provides the mechanism to convert primitive into object and object into primitive. The automatic conversion of primitive into object is known as autoboxing and vice-versa unboxing.

**public** **class** WrapperExample1{

**public** **static** **void** main(String args[]){

//Converting int into Integer

**int** a=20;

Integer i=Integer.valueOf(a);//converting int into Integer

Integer j=a;//autoboxing, now compiler will write Integer.valueOf(a) internally  System.out.println(a+" "+i+" "+j);

}}

**Output:**

20 20 20

**Problem statement:**

1. Reading and writing to a text file
2. Demonstrate wrapper class with suitable example.

**Program:**

**Discussions:**

**Questionnaires:**

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**LAB SESSION 13**

**Title: Java Applet**

**Objective:**

* To make the students understand life cycle of the applets and its functionality.
* To work with GUI environment using applet programming.

**Theory:**

# Java Applet

### An applet is a small application designed to be transmitted over the Internet and executed by a Java-compatible Web browser. Applets are event driven programs. Java applet programs are different from java application.

**Example**

import java.applet.Applet;

import java.awt.Graphics;

public class HelloWorld extends Applet {

public void paint(Graphics g) {

g.drawString("Hello world!", 50, 25);

}

}

There are two methods to run a Java applet program.

    Using Java compatible web browser  
     Using Applet Viewer included with your JDK.

**Using Java compatible web browser**  
  
 Open any text editor like notepad.  
 Type below code

|  |
| --- |
| <applet code = "test" width = 400 height = 300> </applet> |

Save as a .html file in location of your java file.  
     If you want to save your html file in different location,  
        Change above code to:

                  Example:

|  |
| --- |
| <applet code = "D:\my java files\test" width = 400 height = 300> </applet> |

 After successful save, open your html file with any Java compatible web browser.  
 Now you can see your applet in that browser.

**Using Applet Viewer**  
  
 Normally your JDK already contain an Applet Viewer for viewing applets (Search in your jdk folder).  
 For using this method, add this line of code to your java program just before of your class.

|  |
| --- |
| /\* <applet code = "test" width = 300 height = 300> </applet> \*/ |

 Now, totally your program should be like this:

|  |
| --- |
| import java.applet.\*; import java.awt.\*;   /\* <applet code = "test" width = 300 height = 300> </applet> \*/  public class test extends Applet {  public void paint(Graphics g)     {         g.drawString("Hello",123,125);      } } |

 Save again.  
 [Compile your Program.](http://www.infolet.org/2012/08/how-to-make-build-compile-run-java.html)  
 After successful compilation,  type:  **appletviewer test.java**to run your applet.



**Problem statement:**

1. Write an applet to display “Hello World”
2. Draw a filled oval and a filled rectangle.
3. Draw concentric circles using applet.
4. Write an applet to show the text “Welcome” inside an oval.
5. Create an applet to draw a filled square enclosed by two circles.

**Program:**

**Discussions:**

**Questionnaires:**

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**LAB SESSION 14**

**Title: AWT & Event handling**

**Objective**

* To study how to work with event handling.
* GUI creation using Java AWT .

**Theory:**

Java AWT (Abstract Windowing Toolkit) is an API to develop GUI or window-based application in java.Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components uses the resources of system. The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

**Problem statement:**

Accept two numbers from user and display the sum using AWT event handling.

**Program:**

**Discussions:**

**Questionnaires:**

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**LAB SESSION 15**

**Title: Micro Project**

**Objective:** To apply the Java programming concepts in real life situations

**Problem** Projects will be distributed to students in groups of three to four students.

**The assignment covers COs \_\_\_\_\_\_\_\_**

|  |
| --- |
| **Grade awarded:**  **Teacher’s signature with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |