NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India
(ISO 9001:2015 Certified)
Accredited with 'A' Grade by NAAC

Department of Information Science & Engineering

JAVA PROGRAMMING LAB MANUAL

(19IS410)

by

Abhishek S. Rao
Asst. Professor, Dept. of IS&E

for

B.E 4th Semester IS&E

2020 - 21

JAVA PROGRAMMING LABORATORY

Sub Code: 19IS410 **Credits: 01**
Hrs / Week: 0+0+2+0 **Total Hours: 26**

COURSE LEARNING OBJECTIVES

This Course will enable students to

1. To understand the concept of code reusability using inheritance, purpose and creation of packages, implement interfaces and know the exception handling mechanisms in Java.
2. To identify how to create threads and apply concurrency theory in programming
3. To describe how to read and write files from within a Java application.
4. To describe how to use sets, maps, and lists in Java programming.
5. To design and implement Java applications and to understand the fundamentals of event handling in JavaFX.

COURSE CONTENT

	5. Rate of interest (static data) Provide a default constructor and parameterized constructor to this class. Also provide methods: 1. To deposit amount 2. To withdraw amount after checking for minimum balance 3. To display all the details of an account holder 4. Display rate of interest (a static method) Illustrate all the constructors as well as all the methods by defining objects.	
2a.	Create a class called Counter that contains a static data member to count the number of Counter objects being created. Also, define a static member function called showCount() which displays the number of objects created at any given point of time.	**17**
b.	Design a super class called Staff with details as StaffId, Name, Phone, Salary. Extend this class by writing three subclasses namely Teaching (domain, publications), Technical (skills), and Contract (period). Write a Java program to read and display at least 3 staff objects of all three categories.	**18**
3a.	Define a base class called Student with the following two fields: Name, ID. Derive two classes called Sports and Exam from the Student base class. Class Sports has a field called s_grade and class Exam has a field called e_grade which are integer fields. Derive a class called Results which inherit from Sports and Exam. This class has a character array or string field to represent the final result. Also, it has a method called display which can be used to display the final result. Illustrate the usage of these classes in ain method.	**24**
b.	Write a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea( ) that prints the area of the given shape.	**26**
4a.	Write a Java program to implement the concept of importing classes from user defined package and creating packages (Use access modifiers to demonstrate the concept of packages).	**28**

Part - B

Design and implement a Java application using JavaFX for a given problem scenario.

COURSE OUTCOMES

At the end of the course the student will be able to

Sl. No.	Course Outcome	PI	Blooms Taxonomy Level
C410.1	Apply the principles of object-oriented programming for a specific problem.	1.3.1,1.4.1,2.4.4, 3.2.2,4.1.3	L3
C410.2	Design user-defined classes with thread capability and understand the concurrent issues in thread programming	1.3.1,1.4.1,2.4.4, 3.2.2,3.4.2,4.1.3	L3
C410.3	Apply byte streams and character streams for file management	1.3.1,1.4.1,2.4.4, 3.2.2,3.4.2,4.1.3	L3
C410.4	Write programs using the Java Collection API	1.3.1,1.4.1,2.4.4, 3.2.2,3.4.2,4.1.3	L2
C410.5	Develop rich user-interface applications using modern API's such as JavaFX.	1.3.1,1.4.1,2.1.2,2.3.1, 2.4.4,3.2.2,3.4.2, 4.2.1,4.3.1,5.1.1,5.1.2	L4

INSTRUCTIONS TO THE STUDENTS

1. This laboratory manual is an interim record of the sample experiments conductedon programming concepts during the regular laboratory sessions under the above said subject code.
2. Students should come with through preparation for the programs to be developed.
3. Students will not be permitted to attend the laboratory unless they bring the practical record and observation fully completed in all respects pertaining to the experiment conducted in the previous class.
4. Students are supposed to wear their college ID cards before attending the lab.
5. Students are supposed to occupy the systems allotted to them and are not supposed to talk or make noise in the lab.
6. The students are expected to execute / debug the code to fulfill the aim of the experiment.
7. The students need to test the written programs with the varieties of inputs other than the sample inputs and display the output in the record book.
8. Practical record and observation book should be neatly maintained.
9. Students should obtain the signature of the staff-in-charge / Co – in-charge in the observation book after completing each program.

Head-ISE

INTRODUCTION TO JAVA PROGRAMMING

Anyone who is learning to program must choose a programming environment that makes it possible to create and to run programs. Programming environments can be divided into two very different types: integrated development environments and command-line environments. All programming environments for Java require some text editing capability, a Java compiler, and a way to run applets and stand-alone applications. An integrated development environment, or IDE, is a graphical user interface program that integrates all these aspects of programming and probably others (such as a debugger, a visual interface builder, and project management). A command-line environment is just a collection of commands that can be typed in to edit files, compile source code, and run programs.

Command line environment is preferable for beginning programmers. IDEs can simplify the management of large numbers of files in a complex project, but they are themselves complex programs that add another level of complications to the already difficult task of learning the fundamentals of programming.

JAVA was developed by **Sun Microsystems** Inc in the year **1991**, later acquired by Oracle Corporation. It was developed by James Gosling and Patrick Naughton. It is a simple programming language. Java makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs.

Java is a class-based, object-oriented programming language and is designed to have as few implementation dependencies as possible. A general-purpose programming language made for developers to *write once run anywhere* that is compiled Java code can run on all platforms that support Java. Java applications are compiled to byte code that can run on any Java Virtual Machine. The syntax of Java is like c/c++.

Java Terminology

Before learning Java, one must be familiar with these common terms of Java.

1. **Java Virtual Machine(JVM):** This is generally referred to as JVM. There are three execution phases of a program. They are written, compile and run the program.
    - Writing a program is done by java programmer like you and me.
    - The compilation is done by **JAVAC** compiler which is a primary Java compiler included in the Java development kit (JDK). It takes Java program as input and generates bytecode as output.
    - In Running phase of a program, **JVM** executes the bytecode generated by the compiler.

   Now, we understood that the function of Java Virtual Machine is to execute the bytecode produced by the compiler. Every Operating System has different JVM but the output they produce after the execution of bytecode is the same across all the operating systems. Therefore, Java is known as a **platform-independent language.**

2. Bytecode in the **Development process:** As discussed, Javac compiler of JDK compiles the java source code into bytecode so that it can be executed by JVM. It is saved as **.class** file by the compiler. To view the bytecode, a disassembler like javap can be used.

3. Java Development Kit (JDK): While we were using the term JDK, when we learn about bytecode and JVM. So, as the name suggests, it is a complete java development kit that includes everything including compiler, Java Runtime Environment (JRE), java debuggers, java docs etc. For the program to execute in java, we need to install JDK in our computer to create, compile and run the java program.

4. **Java Runtime Environment (JRE):** JDK includes JRE. JRE installation on our computers allow the java program to run, however, we cannot compile it. JRE includes a browser, JVM, applet supports and plugins. For running the java program, a computer needs JRE.

5. Garbage Collector: In Java, programmers can't delete the objects. To delete or recollect that memory JVM has a program called Garbage Collector. Garbage Collector can recollect the of objects that are not referenced. So, Java makes the life of a programmer easy by handling memory management. However, programmers should be careful about their code whether they are using objects that have been used for a long time. Because Garbage cannot recover the memory of objects being referenced.

6. ClassPath: The classpath is the file path where the java runtime and Java compiler looks for **.class** files to load. By default, JDK provides many libraries. If you want to include external libraries, they should be added to the classpath.

Integrated Development Environments

There are sophisticated IDEs for Java programming that are available.

1. **Eclipse IDE --** An increasingly popular professional development environment that supports Java development, among other things. Eclipse is itself written in Java. It is available from http://www.eclipse.org/.

2. **NetBeans IDE --** A pure Java IDE that should run on any system with Java 1.7 or later. NetBeans is a free, "open source" program. It is essentially the open source version of the next IDE. It can be downloaded from www.netbeans.org.

3. **BlueJ --** is a Java IDE written in Java that is meant particularly for educational use. Itis available from http://www.bluej.org/.

4. **JCreator --** for Windows. It looks like a nice lighter-weight IDE that works on top of Sun's SDK. There is a free version, as well as a shareware version. It is available at http://www.jcreator.com. There are other products like JCreator, for Windows and for other operating systems.

Recommended System/Software Requirements

- **System**: Desktop PC with minimum of 2.6GHZ or faster processor with at least 256 MB RAM and 40GB free disk space.
- **Operating system**: Flavor of any WINDOWS.
- **Software**: jdk-12.0.2_windows-x64_bin.
- **Editor**: Eclipse or NetBeans.

Simple Java Program

```
public class MyFirstJavaProgram
{

    public static void main(String [ ]args)
    {
        System.out.println("Hello World");
    }
}
```

javac MyFirstJavaProgram.java

java MyFirstJavaProgram

Output: Hello World

Experiment No: 1a.

Define a class called Customer that holds private fields for a customer ID number, name, and credit limit. Include appropriate constructors to initialize the instance variables of the Customer class. Write a main() method that declares an array of 5 customer objects. Prompt the user for values for each Customer and display all 5 Customer objects.

Program:

```
import java.util.*;
public class Customer
{
    String ID;
    String name;
    int Climit;
    Customer (String ID,String name,int Climit)
    {
          this.ID=ID;
          this.name=name;
          this.Climit=Climit;
    }
    public static void main(String a[])
    {
          Scanner s1=new Scanner(System.in);
          Customer s[]=new Customer [5];
          String ID;
          String name;
          int Climit;
          int n;
          System.out.println("Enter the number of Customer ");
          n=s1.nextInt();

          for(int i=0;i<n;i++)
          {
                System.out.println("Enter Customer Details: ID NAME credit
                limit ");
                ID=s1.next();
                name=s1.next();
                Climit=s1.nextInt();
                s[i]=new Customer(ID,name,Climit);
          }
          System.out.println("Customer Details");
          System.out.println("ID\tNAME\tCLIMIT ");
```

```
        for(int i=0;i<n;i++)
        {

              System.out.println(s[i].ID+"\t"+s[i].name+"\t"+s[i].Climit);

        }
    }
}
```

Experiment No: 1b.

Define a class to represent a Bank Account. Include the following members.

Data Members:

1. **Name of the depositor**
2. **Account number**
3. **Type of account**
4. **Balance amount in the account**
5. **Rate of interest (static data)**

Provide a default constructor and parameterized constructor to this class. Also provide methods:

1. **To deposit amount**
2. **To withdraw amount after checking for minimum balance**
3. **To display all the details of an account holder**
4. **Display rate of interest (a static method)**

Illustrate all the constructors as well as all the methods by defining objects.

Program:

```
import java.util.Scanner;

public class Bank
{
    String name;
    int acno;
    String actype;
    int balance;
    int rate;
    static int n;
    static Bank b[]=new Bank[5];

    Bank()
    {
          System.out.println("Welcome to Bank");
    }

    Bank(String name, int acno,String actype, int balance, int rate)
    {
          this.name=name;
          this.acno=acno;
          this.actype=actype;
          this.balance=balance;
          this.rate=rate;
    }
```

```
    public static void main(String args[])
    {
          String name;
          int acno;
          String actype;
          int balance;
          int rate;
          Scanner sc=new Scanner(System.in);
          System.out.println("Enter the number of accounts you want to
          create");
          n=sc.nextInt();
          new Bank();
          for (int i=0;i<n;i++)
          {
                System.out.println("Enter the Name");
                name=sc.next();
                System.out.println("Enter the Account Number");
                acno=sc.nextInt();
                System.out.println("Enter the Account Type");
                actype=sc.next();
                System.out.println("Enter the Balance Amount");
                balance=sc.nextInt();
                System.out.println("Enter the Rate of Interest");
                rate=sc.nextInt();
                b[i]=new Bank(name,acno,actype,balance,rate);
          }
          for(int i=0;i<n;i++)
          {

                System.out.println(b[i].name+"\n"+b[i].acno+"\n"+b[i].actype
                +"\n"+b[i].balance+"\n"+b[i].rate);
          }
          for(;;)
          {
                System.out.println("Menu");
                System.out.println("1.
                Deposit\n2.Withdraw\n3.Display\n4.Exit");
                int ch=sc.nextInt();
              switch(ch)
              {
                    case 1:
                     System.out.println("Enter the Account Number and amount
                     to deposit");
                     acno=sc.nextInt();
```

```
                        balance=sc.nextInt();
                        deposit(acno,balance);
                        break;
                  case 2:
                    System.out.println("Enter the Account Number and amount
                    to withdraw");
                    acno=sc.nextInt();
                        balance=sc.nextInt();
                        withdraw(acno,balance);
                        break;
                  case 3:disp();
                    break;
                  case 4:
                    System.exit(0);
                    break;
                  default:System.out.println("Invalid Choice");
            }
      }
}
static void deposit(int acno, int bal)
{

      for(int i=0;i<n;i++)
      {
            if(acno==b[i].acno)
            {
                  b[i].balance+=bal;
                  System.out.println(b[i].balance);
            }
            else
            {
                  System.out.println("Not Found");
            }
      }

}
static void withdraw(int acno, int bal)
{

      for(int i=0;i<n;i++)
      {
            if(acno==b[i].acno)
            {
                  b[i].balance-=bal;
```

```
                    System.out.println(b[i].balance);
                }
                else
                {
                    System.out.println("Not Found");
                }
            }
    }
    static void disp()
    {
        System.out.println("Acc_no \t Name \t balance \t Acc_Type \t Rate
          of Interest");
        for(int i=0;i<n;i++)
          System.out.println(b[i].acno+ "\t"+b[i].name+ "\t"+"
          "+b[i].balance+ "\t"+""+b[i].actype + "\t"+""+b[i].rate);
    }
}
```

Experiment No: 2a.

Create a class called Counter that contains a static data member to count the number of Counter objects being created. Also, define a static member function called showCount() which displays the number of objects created at any given point of time.

Program:

```
public class Counter
{
  int objNo;
  static int objCnt;
  Counter()
  {
    objNo = objCnt++;
  }

  static void showCount()
  {
      System.out.println( "count:" + objCnt);
  }
  public static void main(String Ar[])
  {
    Counter t1, t2,t3;
        t1=new Counter();
        Counter.showCount();
        t2=new Counter();
        Counter.showCount();
        t3=new Counter();
        Counter.showCount();
    }
}
```

Experiment No: 2b.

Design a super class called Staff with details as StaffId, Name, Phone, Salary. Extend this class by writing three subclasses namely Teaching (domain, publications), Technical (skills), and Contract (period). Write a Java program to read and display at least 3 staff objects of all three categories.

Program:

```
import java.io.IOException;
import java.util.Scanner;

class Staff
{
	int staffid;
	String name;
	long phone;
	float salary;

	Staff(int staffid, String name, long phone,float salary)
	{
		this.staffid=staffid;
		this.name=name;
		this.phone=phone;
		this.salary=salary;
	}
}

class Teaching extends Staff
{
	String domain;
	String publication;
```

```
    Teaching(int staffid, String name, long phone, float salary, String domain, String publication)
    {
        super(staffid, name, phone, salary);
        this.domain=domain;
        this.publication=publication;
    }

    void display()
    {
        System.out.println(staffid +"\t\t"+ name+"\t\t"+ phone+"\t"+ salary+"\t"+ domain+"\t"+
        publication );
    }
}
class Technical extends Staff
{
    String skills;
    Technical(int staffid, String name, long phone, float salary, String skills)
    {
        super(staffid, name, phone, salary);
        this.skills=skills;
    }
    void display()
    {
        System.out.println(staffid +"\t\t"+ name+"\t\t"+ phone+"\t"+ salary+"\t"+ skills);
    }
}
class Contract extends Staff
{
    int period;
    Contract(int staffid, String name, long phone, float salary, int period)
```

```
        {
                super(staffid, name, phone, salary);
                this.period=period;
        }
        void display()
        {
                System.out.println(staffid +"\t\t"+ name+"\t\t"+ phone+"\t"+ salary+"\t"+ period);
        }
}
public class StaffDetails
{
        public static void main(String argrs[]) throws IOException
        {
                Staff s[]=new Staff[10];
                boolean yes=true;
                int choice;
                Scanner scn=new Scanner(System.in);
                do
                {
                        System.out.println("1).Teaching\n2).Technical\n3).Contract\n4).Exit\n\nEnter
                        Choice");
                        choice = scn.nextInt();
                        switch(choice)
                        {
                                case 1:System.out.println("Enter number of teaching staff details");
                                int n=scn.nextInt();
                                System.out.println("Enter Teaching Staff Details: ID Name Phone Salary
                                Domain Publication");
                                for(int i=0;i<n;i++)
```

```
{
        int id=scn.nextInt();
        String name=scn.next();
        long phone=scn.nextLong();
        float sal=scn.nextFloat();
        String domain=scn.next();
        String publ=scn.next();
        s[i]=new Teaching(id,name,phone,sal,domain,publ);
}

System.out.println("Staff
ID\tName\tPhone\tSalary\tDomain\tPublication");
for(int i=0;i<n;i++)
{
        ((Teaching) s[i]).display();
}
break;
case 2: System.out.println("Enter number of Technical Staff Details");
int t=scn.nextInt();
System.out.println("Enter Technical Staff Details: ID Name Phone Salary
Skill");
for(int i=0;i<t;i++)
{
        int id=scn.nextInt();
        String name=scn.next();
        long phone=scn.nextLong();
        float sal=scn.nextFloat();
        String skill=scn.next();
        s[i]=new Technical(id,name,phone,sal,skill);
}
```

```
                System.out.println("Staff ID\tName\tPhone\tSalary\tSkills");
                for(int i=0;i<t;i++)
                {
                        ((Technical) s[i]).display();
                }
                break;
                case 3: System.out.println("Enter number of Contract Staff Details");
                int c=scn.nextInt();
                System.out.println("Enter Contract Staff Details: ID Name Phone Salary
                Period");
                for(int i=0;i<c;i++)
                {
                        int id=scn.nextInt();
                        String name=scn.next();
                        long phone=scn.nextLong();
                        float sal=scn.nextFloat();
                        int period=scn.nextInt();
                        s[i]=new Contract(id,name,phone,sal,period);
                }
                System.out.println("Staff ID\tName\tPhone\tSalary\tPeriod")
                for(int i=0;i<c;i++)
                {
                        ((Contract) s[i]).display();
                }
                break;
                case 4: scn.close();
                System.exit(0);break;
                default:System.out.println("Invalid Choice");
        }
}while(yes==true);
```

```
    }
}
```

Experiment No: 3a.

Define a base class called Student with the following two fields: Name, ID. Derive two classes called Sports and Exam from the Student base class. Class Sports has a field called s_grade and class Exam has a field called e_grade which are integer fields. Derive a class called Results which inherit from Sports and Exam. This class has a character array or string field to represent the final result. Also, it has a method called display which can be used to display the final result. Illustrate the usage of these classes in an method.

Program:

```
interface Sports
{
     String S_grade="S+";
}

interface Exam
{
    String E_grade="B+";
}

class Student implements Sports,Exam
{
   String Name;
   String ID;
}

public class Result extends Student
{
    Result(String name,String Id)
    {
        Name=name;
        ID=Id;
    }
    void display()
    {
        System.out.println(Name+" "+ ID+" "+S_grade+" "+E_grade);
    }
```

```
    public static void main(String a[])
    {
        Result r=new Result("Ajay","001");
        r.display();
    }
}
```

Experiment No: 3b.

Write a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea( ) that prints the area of the given shape.

Program:

```
import java.util.*;
abstract class Shape
{
        int x,y;
        abstract void printArea(double x,double y);
}
class Rectangle extends Shape
{
        void printArea (double x,double y)
        {
                System.out.println("Area of rectangle is :"+(x*y));
        }
}
class Circle extends Shape
{
        void printArea (double x,double y)
        {
                System.out.println("Area of circle is :"+(3.14*x*x));
        }
}
class Triangle extends Shape
{
        void printArea (double x,double y)
```

```
        {
                System.out.println("Area of triangle is :"+(0.5*x*y));
        }
}
public class AbstactDemo
{
        public static void main(String[] args)
        {
                Rectangle r=new Rectangle();
                r. printArea (2,5);
                Circle c=new Circle();
                c.  printArea  (5,5);
                Triangle t=new Triangle();
                t. printArea (2,5);
        }
}
```

Experiment No: 4a.

Write a Java program to implement the concept of importing classes from user defined packageand creating packages (Use access modifiers to demonstrate the concept of packages).

Program:

```
package mypackage;
public class FirstClass
{
  public String getNames(String s)
  {
      return s;
  }
}

import mypackage.FirstClass;
public class PackageMain
{
   public static void main(String args[])
   {
         FirstClass m=new FirstClass();  //importing classes from user
         defined package
         String name=m.getNames("Santosh");
         System.out.println("Welcome to mypackage "+name);

   }
}
```

Experiment No: 4b.

Design a Stack class. Provide your own stack exceptions namely Push Exception and Pop Exception, which throw exceptions when the stack is full and when the stack is empty respectively. Show the usage of these exceptions in handling a stack object in the main method.

Program:

```
import java.util.*;
class StackException extends Exception
{
    StackException(String s)
    {
          super(s);
   }
}
public class Stack
{
    static  int top=-1;
    static int item[];
    void pushItem(int data)throws StackException
    {
          if (top == item.length - 1)
    {
                throw new StackException("Stack is Full");
    }
          else
          {
                item[++top] = data;
                System.out.println("Pushed Item :" + item[top]);
          }
    }

    void popItem()throws StackException
    {
          if (top < 0)
          {
                throw new StackException("Stack Underflow");
          }
          else
          {
                System.out.println("Pop Item : " + item[top--]);
          }
```

```
    }
    public static void main(String[] args) throws Exception
    {
        int choice;
        Stack st=new Stack();
        Scanner s=new Scanner(System.in);
        System.out.println("Enter size of stack");
        int size=s.nextInt();
        item=new int[size];
        for(;;)
        {
            System.out.println("\n");
            System.out.println("1).Push\n2).Pop\n 3).Exit\n\nEnter
            Choice");
            choice = s.nextInt();
            switch(choice)
            {
                case 1:
                    System.out.println("Enter Push Item: ");
                    st.pushItem(s.nextInt());
                    break;
                case 2:
                    st.popItem();
                    break;
                case 3:
                    System.exit(0);
                    System.out.println("Exit Successfull");
                    break;
                default:
                    System.out.println("Invalid Choice");
            }
        }
    }
}
```

Experiment No: 5a.

Create a class Phone {String brand, int memCapacity}, which contains an interface Callable {makeAudioCall(String cellNum), makeVideoCall(String cellNum)}. Create subclasses BasicPhone and SmartPhone and implement the methods appropriately. Demonstrate the creation of both subclass objects by calling appropriate constructors which accepts value from the user. Using these objects call the methods of the interface.

Program:

```
import java.util.Scanner;

class Phone
{
    String brand;
    int memCapacity;
    interface Callable
    {
          void makeAudioCall(String cellNum);
          void makeVideoCall(String cellNum);
    }
}
class BasicPhone implements Phone.Callable
{
    public void makeAudioCall(String cellNum)
    {
          Phone p=new Phone();
          p.brand="Nokia";
          p.memCapacity=1;
          System.out.println("Basic Phone Audio Call Brand is "+p.brand+
                    " with memory capacity "+p.memCapacity+" GB"+
                    ". Mobile Number is "+cellNum);
    }

    public void makeVideoCall(String cellNum)
    {
          Phone p=new Phone();
          p.brand="Nokia";
          p.memCapacity=1;
          System.out.println("Basic Phone Video Call Brand is "+p.brand+
```

```
                    " with memory capacity "+p.memCapacity+" GB"+
                    ". Mobile Number is "+cellNum);
    }

}
class SmartPhone implements Phone.Callable
{
    public void makeAudioCall(String cellNum)
    {
          Phone p=new Phone();
          p.brand="Samsung";
          p.memCapacity=4;
          System.out.println("Smart Phone Audio Call Brand is "+p.brand+
                    " with memory capacity "+p.memCapacity+" GB"+
                    ". Mobile Number is "+cellNum);
    }

    public void makeVideoCall(String cellNum)
    {
          Phone p=new Phone();
          p.brand="Samsung";
          p.memCapacity=4;
          System.out.println("Smart Phone Video Call Brand is "+p.brand+
                    " with memory capacity "+p.memCapacity+" GB"+
                    ". Mobile Number is "+cellNum);
    }
}
public class MobilePhone
{
    Scanner sc,sc1=null;
    MobilePhone()
    {
          BasicPhone bp=new BasicPhone();
          System.out.println("Enter Basic Phone Cell Number");
          sc=new Scanner(System.in);
          String bpcellnum=sc.next();
          bp.makeAudioCall(bpcellnum);
          bp.makeVideoCall(bpcellnum);

          SmartPhone sp=new SmartPhone();
          System.out.println("Enter Smart Phone Cell Number");
          sc1=new Scanner(System.in);
          String spcellnum=sc1.next();
          sp.makeAudioCall(spcellnum);
```

```
            sp.makeVideoCall(spcellnum);
            sc.close();
            sc1.close();
      }
      public static void main(String args[])
      {
            new MobilePhone();
      }
}
```

Experiment No: 5b.

Write a Java program that implements a multi-thread application that has three threads. First thread generates a random integer for every 1 second; second thread computes the square of the number and prints; third thread will print the value of cube of the number. (Create the threads by extending Thread class and implementing Runnable interface).

Program:

```
//Thread Class
import java.util.Random;
class Square extends
Thread
{
        int x;
        Square(int
        n)
        {
                x = n;
        }
        public void run()
        {
                int sqr = x * x;
                System.out.println("Square of " + x + " = " + sqr );
        }
}
class Cube extends Thread
{
        int x;
        Cube(int n)
        {
                x = n;
```

```
        }
        public void run()
        {
                int cub = x * x * x;
                System.out.println("Cube of " + x + " = " + cub );
        }
}
class Number extends Thread
{
        public void run()
        {
                Random random = new
                Random();for(int i =0; i<10; i++)
                {
                        int randomInteger = random.nextInt(100);
                        System.out.println("Random Integer generated: " + randomInteger);
                        Square s = new Square(randomInteger);
                        s.start();
                        Cube c = new
                        Cube(randomInteger);c.start();
                        try
                        {
                                Thread.sleep(1000);
                        }
                        catch (InterruptedException ex)
                        {
                                System.out.println(ex);
                        }
                }
        }
}
```

```
public class ThreadDemo
{
        public static void main(String args[])
        {
                Number n = new
                Number();n.start();
        }
}
```

//Runnable Interface

```
import java.util.Random;
class SquareNum implements Runnable
{
    public int X;
    public SquareNum(int x)
    {
          X = x;
    }
    public void run()
    {
          System.out.println("Square of " + X + " is: " + X * X);
    }
}
class CubeNum implements Runnable
{
    public int X;
    public CubeNum(int x)
    {
          X = x;
    }
    public void run()
    {
          System.out.println("Cube of " + X + " is: " + X * X * X);
    }
}
class JavaThread implements Runnable
{
    public void run()
    {
          int num = 0;
```

```
        Random r = new Random();
        try
        {
            for (int i = 0; i < 5; i++)
            {
                System.out.println("-------------------------");
                num = r.nextInt(10);
                System.out.println("Random Number Generated is " +
                num);
                SquareNum s=new SquareNum(num);
                Thread s1=new Thread(s);
                s1.start();
                CubeNum c = new  CubeNum(num);
                Thread c1=new Thread(c);
                c1.start();
                Thread.sleep(1000);
            }
        }
        catch (Exception ex)
        {
            System.out.println(ex.getMessage());
        }
    }
    public static void main(String args[])
    {
        JavaThread j=new JavaThread();
        Thread j1=new Thread(j);
        j1.start();
    }
}
```

Experiment No: 6a.

Write a Java program to copy contents of input.txt to output.txt using byte stream class and character stream classes.

Program:

```
import java.io.*;
public class FileByteStream
{
    public static void main(String args[]) throws IOException
    {
          FileInputStream in = null;
          FileOutputStream out = null;
          try
          {
                in = new FileInputStream("D:\\Eclipse\\Java
                Lab\\src\\input.txt");
                out = new FileOutputStream("D:\\Eclipse\\Java
                Lab\\src\\output.txt");
                int c;
                while ((c = in.read()) != -1)
                {
                      out.write(c);
                }
                System.out.println("File copied successfully");
          }
          catch(Exception e)
          {
                System.out.println(e);
          }
    }
}
```

```
import java.io.*;
class FileCharacterStream
{
  public static void main(String args[])
  {
    FileReader fr = null;
    FileWriter fw = null;
    try
    {
```

```
            fr=new FileReader("D:\\Eclipse\\Java Lab\\src\\input.txt");
            fw=new FileWriter("D:\\Eclipse\\Java Lab\\src\\output.txt");
            int c=fr.read();
            while(c!=-1)
            {
                  fw.write(c);
                  c = fr.read();
            }
            System.out.println("File copied successfully");
      }
      catch(IOException e)
      {
            System.out.println(e);
      }
      finally
      {
          if (fr != null)
          {
              try
              {
                  fr.close();
              }
              catch(IOException e)
              {

              }
          }
          if (fw != null)
          {
              try
              {
                  fw.close();
              }
              catch(IOException e)
              {

              }
           }
        }
    }
}
```

Experiment No: 6b.

Write a Java program to illustrate the concept of inter thread communication.

Program:

```
class Customer
{
	int amount=10000;
	synchronized void withdraw(int amount)
	{
		System.out.println("going to withdraw...");
		if(this.amount<amount)
		{
			System.out.println("Less balance; waiting for deposit...");
			try
			{
				wait();
			}
			catch(Exception e){}
		}
		this.amount-=amount;
		System.out.println("withdraw completed...");
	}
	synchronized void deposit(int amount)
	{
		System.out.println("going to deposit...");
		this.amount+=amount;
		System.out.println("deposit completed... ");
		notify();
	}
}
```

```
class InterThreadDemo
{
	public static void main(String args[])
	{
		final Customer c=new Customer();
		new Thread()
		{
			public void run()
			{
				c.withdraw(15000);
			}
		}.start();
		new Thread()
		{
			public void run()
			{
				c.deposit(10000);
			}
		}.start();
	}
}
```

Alternate Program

```
public class InterThread2
{

	public static void main(String[] args) throws InterruptedException
	{
		ThreadB b = new ThreadB();
		b.start();
		synchronized (b)
		{
```

```
                System.out.println("main thread calling wait() method"); //
                step 1
                b.wait();
                System.out.println("main thread got notification call"); //
                step 4
                System.out.println("total balance " + b.totalBalance);
          }
     }
}

class ThreadB extends Thread
{
     int totalBalance = 0;
     public void run()
     {
          synchronized (this)
          {
                System.out.println("child Thread starts calculation for
                total balance"); // step 2
                for (int i = 0; i <= 5; i++)
                {
                     totalBalance = totalBalance + i;
                }
                System.out.println("child thread gives notification call");
                // step 3
                this.notify();
          }
     }
}
```

Experiment No: 7a.

Write a JavaFX application program that handles the event generated by Button control.

Program:

(Refer https://www.javatpoint.com/javafx-with-eclipse for installation of JavaFX in Eclipse)

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.*;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.control.Label;
import javafx.stage.Stage;
public class JavaButton extends Application
{

   // launch the application
   public void start(Stage s)
   {
       s.setTitle("creating buttons");

       Button b = new Button("Button");

       TilePane r = new TilePane();

       Label l = new Label("button not selected");

       // action event
          EventHandler<ActionEvent> event = new EventHandler<ActionEvent>()
          {
                public void handle(ActionEvent e)
                {
                      l.setText("   button   selected    ");
                }
          };

       // when button is pressed
       b.setOnAction(event);

       // add button
       r.getChildren().add(b);
```

```
        r.getChildren().add(l);

        // create a scene
        Scene sc = new Scene(r, 200, 200);

        // set the scene
        s.setScene(sc);

        s.show();
    }

    public static void main(String args[])
    {
        // launch the application
        launch(args);
    }
}
```

Experiment No: 7b.

Write a Java program to demonstrate positional access operations, search operations, subList operations on List interface.

Program:

```
/** Positional Access*/
import java.util.*;
public class ListDemo
{
	public static void main(String[] args)
	{
		// Creating a list
		List<Integer> l1 = new ArrayList<Integer>();
		l1.add(0, 1); // adds 1 at 0 index
		l1.add(1, 2); // adds 2 at 1 index
		System.out.println(l1); // [1, 2]
		List<Integer> l2 = new ArrayList<Integer>();
		l2.add(1);
		l2.add(2);
		l2.add(3);
		l1.addAll(1, l2);
		System.out.println(l1);
		l1.remove(1);
		System.out.println(l1); // [1, 2, 3, 2]
		// Prints element at index 3
		System.out.println(l1.get(3));
		// Replace 0th element with 5
		l1.set(0, 5);
		System.out.println(l1);
	}
```

```
}
```

/ Search Operations*/**

```
import java.util.*;
public class ListDemo
{
	public static void main(String[] args)
	{
		// Type safe array list, stores only string
		List<String> l = new ArrayList<String>(15);
		l.add("NMAMIT");
		l.add("NITTE");
		l.add("UDUPI DISTRICT");
		System.out.println("first index of Geeks:" + l.indexOf("NMAMIT"));
		System.out.println("last index of Geeks:" + l.lastIndexOf("UDUPI DISTRICT"));
		System.out.println("Index of element" + " not present : " + l.indexOf("NET"));
	}
}
```

/ subList Operations*/**

```
import java.util.*;
public class ListDemo
{
	public static void main(String[] args)
	{
		// Type safe array list, stores only string
		List<String> l = new ArrayList<String>(5);
		l.add("NMAMIT");
		l.add("NITTE");
		l.add("UDUPI");
```

```
                l.add("DISTRICT");
                l.add("KARNATAKA");
                List<String> range = new ArrayList<String>();
                // Return List between 2nd(including)
                // and 4th element(excluding)
                range = l.subList(2, 4);
                System.out.println(range);
        }
}
```

Experiment No: 8a.

Design a simple calculator to show the working for simple arithmetic operations using Java FX. Use GridLayout to design the layout.

Program:

Refer the following website and design the calculator.

1. https://www.javatpoint.com/javafx-with-eclipse

2. https://download.java.net/general/javafx/eclipse/tutorial.html

3. https://www.youtube.com/watch?v=Q_1cZYoGoYM
   JavaFx Tutorial For Beginners 8 - How to build a Calculator in JavaFX Part-1

Experiment No: 8b.

Write a Java program to illustrate the concept of HashMap, TreeMap, and LinkedHashMap.

Program:

```
import java.util.*;

public class HashMapProg
{
    public static void main(String args[])
    {
      HashMap<Integer,String> map=new HashMap<Integer,String>();
      map.put(1,"SANTOSH");
      map.put(2,"ABHI");
      map.put(3,"AMAR");
      map.put(4,"VIJAY");
      System.out.println("Iterating Hashmap...");
      for(Map.Entry m : map.entrySet())
      {
        System.out.println(m.getKey()+" "+m.getValue());
      }

      TreeMap<Integer,String> map1=new TreeMap<>();
      map1.put(2,"SANTOSH");
      map1.put(3,"ABHI");
      map1.put(1,"AMAR");
      map1.put(4,"VIJAY");
      System.out.println("Iterating Treemap...");
      for(Map.Entry m1 : map1.entrySet())
      {
        System.out.println(m1.getKey()+" "+m1.getValue());
      }

      LinkedHashMap<Integer,String> map2=new
          LinkedHashMap<Integer,String>();
      map2.put(4,"SANTOSH");
      map2.put(3,"ABHI");
      map2.put(2,"AMAR");
      map2.put(1,"VIJAY");
      System.out.println("Iterating LINKED HASHMAP...");
      for(Map.Entry m2:map2.entrySet())
      {
          System.out.println(m2.getKey()+" "+m2.getValue());
```

```
        }
    }
}
```

TEXT BOOK:

1. Java™: The Complete Reference, Seventh Edition, Herbert Schildt, Tata McGraw Hill, 2007.
2. Jim Keogh: J2EE-TheCompleteReference, McGraw Hill, 2007

REFERENCE BOOKS:

1. Y. Daniel Liang: Introduction to JAVA Programming, 7thEdition, Pearson Education, 2007.
2. Stephanie Bodoff et al: The J2EE Tutorial, 2nd Edition, Pearson Education,2004.
3. Uttam K Roy, Advanced JAVA programming, Oxford University press, 2015.
4. E Balagurusamy, Programming with Java A primer, Tata McGraw Hill companies.
5. Anita Seth and B L Juneja, JAVA One step Ahead, Oxford University Press, 2017.

E-RESOURCES

1. NOC:Programming in Java – Nptel
2. https://www.codecademy.com/learn/learn-java
3. https://www.javaworld.com/blog/java-101/
4. https://docs.oracle.com/javase/tutorial/index.html
5. https://www.sololearn.com/Course/Java/
6. https://examples.javacodegeeks.com/

SAMPLE VIVA QUESTIONS

1. What is Java?
2. What are the differences between C++ and Java?
3. List the features of Java Programming language.
4. What do you understand by Java virtual machine?
5. What is the difference between JDK, JRE, and JVM?
6. What is JIT compiler?
7. What is the platform?
8. What are the main differences between the Java platform and other platforms?
9. What gives Java its 'write once and run anywhere' nature?
10. What if I write static public void instead of public static void?
11. What is the default value of the local variables?
12. What are the various access specifiers in Java?
13. What is the purpose of static methods and variables?
14. What are the advantages of Packages in Java?
15. What is an object?
16. What will be the initial value of an object reference which is defined as an instance variable?
17. What is the constructor?
18. How many types of constructors are used in Java?
19. Is constructor inherited?
20. Can you make a constructor final?
21. Can we overload the constructors?
22. What are the differences between the constructors and methods?
23. What is the static method?
24. Why is the main method static?
25. What is the static block?
26. Can we execute a program without main() method?
27. What is this keyword in java?
28. Can we assign the reference to this variable?

29. What is the Inheritance?
30. Which class is the superclass for all the classes?
31. Why is multiple inheritance not supported in java?
32. What is super in java?
33. What are the differences between this and super keyword?
34. What is method overloading?
35. Can we overload the main() method?
36. What is the final variable?
37. What is an Exception?
38. What is a Thread?
39. What is an Interface?
40. What are collections in Java?