JAVA Laboratory [As per Choice Based Credit System (CBCS) scheme] SEMESTER – I			
Subject Code	20MCA107L	CIE Marks	50
Number of Lecture Hours/Week	02 Hrs Laboratory	SEE Marks	50
		SEE Hours	03
CREDITS – 02			

Course Outcomes(CO):

This laboratory course enable students to get practical experience in design, develop, implement, analyze and evaluation/testing of

CO1: Understand Java programming language fundamentals and run time environment.
CO2: Gain knowledge and skill necessary to write java programs.
CO3: Learn the object oriented concepts and its implementation in Java.
CO4: Implement the multithreading and client side programming.
CO5: To understand the working procedure of JDBC

Laboratory Experiments:

SECTION A

1.
a. Write a JAVA Program to demonstrate Constructor Overloading and Method Overloading.
b. Write a JAVA Program to implement Inner class and demonstrate its Access protection.

2.
Write a program in Java for String handling which performs the following:
i) Checks the capacity of StringBuffer objects.
ii) Reverses the contents of a string given on console and converts the resultant string in upper case.
iii) Reads a string from console and appends it to the resultant string of ii.

3.
a. Write a JAVA Program to demonstrate Inheritance.
b. Simple Program on Java for the implementation of Multiple inheritance using interfaces to calculate the area of a rectangle and triangle.

4.
Write a JAVA program which has
i. A Class called Account that creates account with 500Rs minimum balance, a deposit() method to deposit amount, a withdraw() method to withdraw amount and also throws LessBalanceException if an account holder tries to withdraw money which makes the balance become less than 500Rs.
ii. A Class called LessBalanceException which returns the statement that says withdraw amount (Rs) is not valid.
iii. A Class which creates 2 accounts, both account deposit money and one account tries to withdraw more money which generates a LessBalanceException take appropriate action for the same.

5. Write a JAVA program using Synchronized Threads, which demonstrates Producer Consumer concept.

6. Write a JAVA program to implement a Queue using user defined Exception Handling (also make use of throw, throws.).
7. Complete the following: 1. Create a package named shape. 2. Create some classes in the package representing some common shapes like Square,Triangle, and Circle. 3. Import and compile these classes in other program.
8. Write a JAVA Program a. Create an enumeration Day of Week with seven values SUNDAY through SATURDAY. Add a method is Workday( ) to the DayofWeek class that returns true if the value on which it is called is MONDAY through FRIDAY. For example, the call DayOfWeek.SUNDAY.isWorkDay ( ) returns false.
9. Write a JAVA program which has i. A Interface class for Stack Operations ii. A Class that implements the Stack Interface and creates a fixed length Stack. iii. A Class that implements the Stack Interface and creates a Dynamic length Stack. iv. A Class that uses both the above Stacks through Interface reference and does the Stack operations that demonstrates the runtime binding.
10. Write a JAVA program to print a chessboard pattern.
11. Write a JAVA Program which uses FileInputStream / FileOutPutStream Classes.
12. Write JAVA programs which demonstrates utilities of LinkedList Class.
13. Write a JAVA program to implement JDBC operations.

Note: In the examination each student should do one question out of the above 13 questions and One of examiners choice.

1.

a. Write a JAVA Program to demonstrate Constructor Overloading and Method Overloading.

```
class Test1a
{
  int a;

  Test1a()
  {
    a=1;
  }
  Test1a(int i)
  {
    a=i;
  }

 void show()
   {
    System.out.println("\n Value of a : " +a);
   }

 void show(int m)
 {
  System.out.println("\n Value of argument passed : " +m);
 }
}

class Q1a
{
 public static void main(String args[])
 {
   Test1a ob1=new Test1a();
   Test1a ob2=new Test1a(10);

   ob1.show();
   ob2.show();
   ob2.show(33);

 }
}
```

Output:

```
Value of a: 1
Value of argument passed: 10
Value of argument passed: 33
```

2. Write JAVA Program to implement Inner class and demonstrate its Access Protections.

```
class Outer
{
        int x=100;
        int y=10;
        void test()
        {
                Inner ob1=new Inner();
                ob1.display();
        }
        class Inner
        {
          int z;
          Inner()
           {
              y=90;
              z=60;
           }
        void display()
                {       System.out.println("Display : x = " +x);
                System.out.println("Display : y =" +y);
                System.out.println("Display : z = " +z);
                }
        }//Inner class closes here
```

```
void show()
        {
            System.out.println("Display : x = " +x);
                System.out.println("Display : y = " +y);
}
 }//outer class closes here


 class Q1b
 {
        public static void main(String args[])
        {
                Outer ob1=new Outer();
                ob1.test();
                ob1.show();
        }
 }
```

Output:

```
Display : x = 100
Display : y =90
Display : z = 60
Display : x = 100
Display : y = 90
```

2. Write a program in Java for String handling which performs the following:

i) Checks the capacity of StringBuffer object.
ii) Reverse the contents of a string given on console and converts the resultant string in upper case.
iii) Reads a string from console and append it to the resultant string of ii.

i)

```
public class StringBuf
{
   public static void main(String args[ ] )
   {
      StringBuffer buffer1 = new StringBuffer( ) ;
      StringBuffer buffer2 = new StringBuffer(50) ;
      StringBuffer buffer3 = new StringBuffer("hello") ;
      System.out.println("buffer1 capacity: " + buffer1.capacity());
      System.out.println("buffer2 capacity: " + buffer2.capacity());
      System.out.println("buffer3 capacity: " + buffer3.capacity());

      System.out.println("\nbuffer1 length: " + buffer1.length());
      System.out.println("buffer2 length: " + buffer2.length());
      System.out.println("buffer3 length: " + buffer3.length());
      buffer3.ensureCapacity(150);
      System.out.println("After modifying buffer3 capacity: " + buffer3.capacity());
   }
}
```

Output:

buffer1 capacity: 16

buffer2 capacity: 50

buffer3 capacity: 21

buffer1 length: 0

buffer2 length: 0

buffer3 length: 5

After modifying buffer3 capacity: 150

ii) And iii)

```
import java.util.Scanner;
class Two_a
{
  public static void main(String args[])
  {
    String original, reverse = "",appnd;
    Scanner in = new Scanner(System.in);
    System.out.println("Enter a string to reverse : ");
    original = in.nextLine();
    int length = original.length();
    for ( int i = length - 1 ; i >= 0 ; i-- )
     reverse = reverse + original.charAt(i);
     System.out.println("Reverse of entered string is: "+reverse);
     System.out.println("String in Upper case: "+reverse.toUpperCase());
     System.out.println("Enter a string to appends: ");
     appnd = in.nextLine();
     System.out.println("After Appending: "+reverse.toUpperCase().concat(appnd));
  }
}
```

Output:

```
Reverse of entered string is: tnemtrapeD
String in Upper case: TNEMTRAPED
After Appending: TNEMTRAPEDMCA
```

3 a. Write a JAVA Program to demonstrate Inheritance.

```
class Parent
{
 int x;
 Parent(int a)
 {
   x=a;
 }
 void displayx()
 {
  System.out.println("\n Value of i : " +x);
 }
}

class Child extends Parent
{
 int y;
 Child(int a,int b)
 {
   super(a);
   y=b;
 }
 void displayxy()
 {
  System.out.println("\n Value of a : " +x);
  System.out.println("\n Value of b : " +y);
 }

}
class Inheritance
{
 public static void main(String args[])
 {
     Parent ob1=new Parent(10);
     Child  ob2=new Child(20,30);

     System.out.println("\n Contents of Parent or Super Class Object : ");
     ob1.displayx();
     System.out.println("\n Contents of Child Class Object : ");
     ob2.displayxy();
  }
}
```

Output:

Contents of Parent or Super Class Object :
Value of i : 10
Contents of Child Class Object :
Value of a : 20
Value of b : 30

b. Simple Program on Java for the implementation of Multiple inheritance using interfaces to calculate the area of a rectangle and triangle.

```
interface Shape{
void area();
} // end of interface
class Rectangle implements Shape{
double l,b;
Rectangle(double length, double breadth){
l=length;
b=breadth;
}
public void area(){
System.out.println("Area of Rectangle is : " + l*b);
}
}
class Triangle implements Shape{
double b,h;
Triangle (double base, double height){
b= base;
h= height;
}
public void area(){
System.out.println("Area of Triangle is : " + (b*h/2));
}
}
public class InterfaceDemo {
  public static void main(String args[]){
  Rectangle rect=new Rectangle(10,05);
rect.area();

Triangle tri=new Triangle(10,20);
tri.area();

  }// end of main
} //end of InterfaceDemo
```

Output:

Area of Rectangle is: 50.0
Area of Triangle is: 100.0

4.

Write a JAVA program which has
i. A Class called Account that creates account with 500Rs minimum balance, a deposit() method to deposit amount, a withdraw() method to withdraw amount and also throws LessBalanceException if an account holder tries to withdraw money which makes the balance become less than 500Rs.
ii. A Class called LessBalanceException which returns the statement that says withdraw amount (Rs) is not valid.
iii. A Class which creates 2 accounts, both account deposit money and one account tries to withdraw more money which generates a LessBalanceException take appropriate action for the same.

```
class Account
{
    int bal=500;
    void deposit(int amt)
    {
        bal+=amt;
    }
    void withdraw(int amt)
    {
        if ((bal-amt)<=500)
        {
            try{
                throw new LessBalanceException(amt);
              }catch (LessBalanceException e1)
              {  }
        }else
            bal-=amt;
    }
}
class LessBalanceException extends Exception
{

    LessBalanceException(int amt)
    {
        System.out.println("Withdrawing of "+amt+"  not possible");
    }
}
public class Q3 {
    public static void main(String[] args)
    {
```

```
    Account ob1=new Account();
    Account ob2=new Account();
    ob1.deposit(200);
    ob2.deposit(200);
    ob2.withdraw(300);
    ob2.withdraw(100);
    System.out.println("Current Balance for ob1 = "+ob1.bal);
    System.out.println("Current Balance for ob2 = "+ob2.bal);
   }
}
```

Output:

```
Withdrawing of 300  not possible
Current Balance for ob1 = 700
Current Balance for ob2 = 600
```

5. Write a JAVA program using Synchronized Threads, which demonstrates Producer Consumer concept.

```
import java.lang.*;
import java.io.*;
import java.util.*;

class common
{
 boolean flag=false;
 String str;
 public  synchronized void produce()throws Exception
  {
    if(flag==false)
        {
          Scanner sc=new Scanner(System.in);
          System.out.println("enter the string");
          str=sc.next();
          flag=true;
          notify();
        }
    else
          wait();
   }
   public synchronized void consume() throws Exception
     {
      if(flag==true)
       {
         System.out.println("the produced string by producer is : "+str);
         flag=false;
         notify();
       }
      else  wait();

     }
}
```

```
class producer extends Thread
{
  common c;
  producer(common c)
     { this.c=c; }

  public void run()
    {
       try
          {
           c.produce();
          }catch(Exception e){}
     }
 }
class consumer extends Thread
{
    common c;
    consumer(common c)
    {     this.c=c;  }
    public void run()
     {
       try
          { c.consume();  }catch(Exception e){}
      }
 }

public class pc
{
  public static void main(String args[]) throws Exception
   {
        common c=new common();
        producer p=new producer(c);
        consumer co=new consumer(c);
        p.start();
        co.start();
    }
}
```

Output:

Enter the string :MCA

the produced string by producer is : MCA

6. Write a JAVA program to implement a Queue using user defined Exception Handling (also make use of throw, throws.).

```
import java.util.Scanner;
class ExcQueue extends Exception
{
ExcQueue(String s)
{
super(s);
}
}
class Queue
 {
int front,rear;
int q[ ]=new int[10];
Queue()
{
rear=-1;
front=-1;
}
void enqueue(int n) throws ExcQueue
{
if (rear==9)throw new ExcQueue("Queue is full");
rear++;
q[rear]=n;
if (front==-1)front=0;
}
int dequeue() throws ExcQueue
{
if (front==-1)throw new ExcQueue("Queue is empty");
int temp=q[front];
if (front==rear)
front=rear=-1;
  else
front++;
return(temp);
}
}
class UseQueue
 {
```

```
public static void main(String args[ ])
{
Queue a=new Queue();
try {
a.enqueue(5);
a.enqueue(20);
}
 catch (ExcQueue e)
{
System.out.println(e.getMessage());
}
try {
System.out.println(a.dequeue());
System.out.println(a.dequeue());
System.out.println(a.dequeue());
}
catch(ExcQueue e)
{
System.out.println(e.getMessage());
}
}
}
```

Output:

5
20
Queue is empty

7. Complete the following:
1. Create a package named shape.
2. Create some classes in the package representing some common shapes like Square,Triangle, and Circle.
3. Import and compile these classes in other program.

Create a package, Shape separately for each……

```
package shape;

public class Circle {

    private double r, PI=3.14;
    public Circle(double radius)
     {
    r=radius;
     }
    public void area(){
    System.out.println("Area of circle ......."+(PI*r*r));
     }

}
```

```
package shape;
public class Square
{
    private double a;
    public Square(double side)
     {
    a=side;
     }
    public void area(){
    System.out.println("Area of Square ......."+(4*a));
     }

}
```

```
package shape;
public class TriangleOne
{
    private double b,h;
    public TriangleOne(double base, double height)
     {
    b=base; h=height;
     }
   public void area(){
```

```
    System.out.println("Area of Triangle ......."+(b*h / 2));
     }

}
```

```
import shape.*;
public class Lab7 {
    public static void main(String arg[]){
     Circle c=new Circle(20);
     c.area();
     Square s=new Square(15);
     s.area();
    TriangleOne obj = new TriangleOne(12,6);
    obj.area();
     }
}
```

Output:

Area of Circle is : 1256.0

Area of Square 60.0

Area of Triangle 36.0

8. Write a JAVA Program

a. Create an enumeration Day of Week with seven values SUNDAY through SATURDAY. Add a method is Workday( ) to the DayofWeek class that returns true if the value on which it is called is MONDAY through FRIDAY. For example, the call DayOfWeek.SUNDAY.isWorkDay ( ) returns false.

```
public class DayOfWeek{
   public enum Day{
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
    THURSDAY, FRIDAY, SATURDAY
}
   Day day;
   public DayOfWeek(Day day) {
   this.day = day;
   System.out.println(day);
    }
   public boolean isWorkday(){
      if ((day == Day.MONDAY)||(Day.TUESDAY==day)||(day==Day.WEDNESDAY) ||
          (day==Day.THURSDAY) || (day==Day.FRIDAY))
      return true;
       else
      return false;
    }
   public static void main(String[] args) {
       DayOfWeek firstDay = new DayOfWeek(Day.FRIDAY);
      boolean a =firstDay.isWorkday();
      System.out.println("returned value is "+ a);
       DayOfWeek sixthDay = new DayOfWeek(Day.SUNDAY);
      boolean b=sixthDay.isWorkday();
      System.out.println("returned value is "+ b);
    }
}
```

Output:

FRIDAY
returned value is true
SUNDAY
returned value is false

9.

Write a JAVA program which has

i. A Interface class for Stack Operations

ii. A Class that implements the Stack Interface and creates a fixed length Stack.

iii. A Class that implements the Stack Interface and creates a Dynamic length Stack.

iv. A Class that uses both the above Stacks through Interface reference and does the Stack operations that demonstrates the runtime binding.

```
import java.util.*;

interface mystackinterface
{
void push(int a);
int pop();
boolean isempty();
}
class fixedstack implements mystackinterface
{
int top; int st[]; fixedstack()
{
top=-1;
st=new int[10];
}
public boolean isempty()
{
if (top==-1)
return true;
else
return false;
}
public void push(int a)
{
if(top!=9)
{
st[++top]=a;
System.out.println(a+" Added to FIXED LENGTH STACK");
}
else
System.out.println("Fixed Length Stack full");
}
```

```
public int pop()
{
return st[top--];
}
}
class dynamictack implements mystackinterface
{
int top; ArrayList st; dynamictack()
{
top=-1;
st=new ArrayList();
}
public boolean isempty()
{
if (top==-1)
return true;
else
return false;
}

public void push(int a)
{
top++;
st.add(a);
}
public int pop()
{
Integer ob=(Integer)st.remove(top--);
return ob.intValue();
}
}
public class mystackimpl
{
public static void main(String[] args)
{mystackinterface fstk=new fixedstack();
mystackinterface dstk=new dynamictack();
for(int i=0;i<15;i++)
fstk.push(i);
for(int i=0;i<15;i++)
if(!fstk.isempty())
```

```
System.out.println("Top Element of Fixed Length Stack : "+fstk.pop());
else
System.out.println("FIXED LENGTH STACK IS EMPTY");
for(int i=0;i<15;i++)
{
dstk.push(i);
System.out.println(i+" Added to Dynamic  LENGTH STACK");
}
for (int i = 0; i < 15; i++)
if(!dstk.isempty())
System.out.println("Top Element of Dynamic Length Stack : "+dstk.pop());
else
System.out.println("Dynamic LENGTH STACK IS EMPTY");
}
}
```

Output:

```
0 Added to FIXED LENGTH STACK
1 Added to FIXED LENGTH STACK
2 Added to FIXED LENGTH STACK
3 Added to FIXED LENGTH STACK
4 Added to FIXED LENGTH STACK
5 Added to FIXED LENGTH STACK
6 Added to FIXED LENGTH STACK
7 Added to FIXED LENGTH STACK
8 Added to FIXED LENGTH STACK
9 Added to FIXED LENGTH STACK
Fixed Length Stack full
Fixed Length Stack full
Fixed Length Stack full
Fixed Length Stack full
Fixed Length Stack full
Top Element of Fixed Length Stack : 9
Top Element of Fixed Length Stack : 8
Top Element of Fixed Length Stack : 7
Top Element of Fixed Length Stack : 6
Top Element of Fixed Length Stack : 5
Top Element of Fixed Length Stack : 4
Top Element of Fixed Length Stack : 3
Top Element of Fixed Length Stack : 2
Top Element of Fixed Length Stack : 1
```

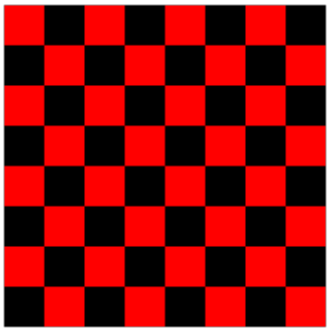
```
Top Element of Fixed Length Stack : 0
FIXED LENGTH STACK IS EMPTY
FIXED LENGTH STACK IS EMPTY
FIXED LENGTH STACK IS EMPTY
FIXED LENGTH STACK IS EMPTY
FIXED LENGTH STACK IS EMPTY
0 Added to Dynamic  LENGTH STACK
1 Added to Dynamic  LENGTH STACK
2 Added to Dynamic  LENGTH STACK
3 Added to Dynamic  LENGTH STACK
4 Added to Dynamic  LENGTH STACK
5 Added to Dynamic  LENGTH STACK
6 Added to Dynamic  LENGTH STACK
7 Added to Dynamic  LENGTH STACK
8 Added to Dynamic  LENGTH STACK
9 Added to Dynamic  LENGTH STACK
10 Added to Dynamic  LENGTH STACK
11 Added to Dynamic  LENGTH STACK
12 Added to Dynamic  LENGTH STACK
13 Added to Dynamic  LENGTH STACK
14 Added to Dynamic  LENGTH STACK
Top Element of Dynamic Length Stack : 14
Top Element of Dynamic Length Stack : 13
Top Element of Dynamic Length Stack : 12
Top Element of Dynamic Length Stack : 11
Top Element of Dynamic Length Stack : 10
Top Element of Dynamic Length Stack : 9
Top Element of Dynamic Length Stack : 8
Top Element of Dynamic Length Stack : 7
Top Element of Dynamic Length Stack : 6
Top Element of Dynamic Length Stack : 5
Top Element of Dynamic Length Stack : 4
Top Element of Dynamic Length Stack : 3
Top Element of Dynamic Length Stack : 2
Top Element of Dynamic Length Stack : 1
Top Element of Dynamic Length Stack : 0
```

10. Write a JAVA program to print a chessboard pattern.

```
import java.awt.*;
   import java.applet.*;
```

```
public class Checkerboard extends Applet {
   public void paint(Graphics g) {
     int row;   // Row number, from 0 to 7
     int col;   // Column number, from 0 to 7
     int x,y;   // Top-left corner of square
     for ( row = 0;  row < 8;  row++ ) {
       for ( col = 0;  col < 8;  col++) {
         x = col * 20;
         y = row * 20;
         if ( (row % 2) == (col % 2) )
           g.setColor(Color.red);
         else
           g.setColor(Color.black);
         g.fillRect(x, y, 20, 20);
       }
     } // end for row
   }  // end paint()
}  // end class
```

Output:



11. Write a JAVA Program which uses FileInputStream / FileOutPutStream Classes.

```
import java.io.*;
public class q8
{
```

```
    public static void main(String[] args)
    {
      try{
           int count=0;
        FileInputStream fis=new FileInputStream("/abc.txt");
        int avail=fis.available();
        byte []b=new byte[avail]; int
        done=fis.read(b);
        System.out.println("File Contents");
        for(byte a:b)
                {
                        Char  i=(char)a;
                        System.out.print((char)a+"");
                 }
        FileOutputStream fos=new FileOutputstream("/xyz.txt");
        fos.write(b);
         }
        catch(Exception e){}
    }
}
```

12. Write JAVA programs which demonstrates utilities of LinkedList Class

```
import java.util.*;

class Q7
 {
```

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

      LinkedList ll = new LinkedList();

      ll.add("F");
      ll.add("B");
      ll.add("D");
      ll.add("E");
      ll.add("C");
      ll.addLast("Z");
      ll.addFirst("A");
      ll.add(1, "A2");
      System.out.println("Original contents of linked list: " + ll);
      System.out.println("Index of first element " +  ll.indexOf("A"));
      ll.remove("F");
      ll.remove(2);
      System.out.println("Contents of linked list after deletion: " + ll);
      ll.removeFirst();
      ll.removeLast();
      System.out.println("linked list after deleting first and last: " + ll);
   // get and set a value
      Object val = ll.get(2);
      ll.set(2, "omega");
      System.out.println("linked List after change: " + ll);
    }
}
```

Output:

```
Original contents of linked list: [A, A2, F, B, D, E, C, Z]
Index of first element 0
Contents of linked list after deletion: [A, A2, D, E, C, Z]
linked list after deleting first and last: [A2, D, E, C]
linked List after change: [A2, D, E, C]
```

13. Write a JAVA program to implement JDBC operations.

```
import java.io.*;
import java.sql.*;

class GFG {
  public static void main(String[] args) throws Exception
  {
    String url= "jdbc:mysql://localhost:3306/table_name"; // table details
    String username = "rootgfg"; // MySQL credentials
    String password = "gfg123";
    String query= "select *from students"; // query to be run
    Class.forName("com.mysql.cj.jdbc.Driver"); // Driver name
    Connection con = DriverManager.getConnection(url, username, password);
    System.out.println("Connection Established successfully");
    Statement st = con.createStatement();
    ResultSet rs = st.executeQuery(query); // Execute query
    rs.next();
    String name= rs.getString("name"); // Retrieve name from db
    System.out.println(name); // Print result on console
    st.close(); // close statement
    con.close(); // close connection
    System.out.println("Connection Closed....");
  }
}
```

Output: