DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

LAB MANUAL

CS6461

OBJECT ORIENTED PROGRAMMING LABORATORY

Ex: No: 1(a) FUNCTION OVERLOADING

Aim:

To write a c ++ program to implement the function overloading

Algorithm:

```
Step 1: Start a program.
Step 2: Declare the function variable.
Step 3: Invoke the main function in that of the set and value.
Step 4: Declare the function area.
Step 5: Display the data given.
Step 6: Stop the program.
```

Program:

```
#include
#include
int volume(int);
doublevolume(int);
long volume(long,int,int);
int main()
{
clrscr();
cout<<volume(10)<<endl;< span=""></volume(10)<<endl;<>
cout<<volume(2,5,8)<<endl;< span=""></volume(2,5,8)<<endl;<>
cout<<volume(100,75,15)<<endl;< span=""></volume(100,75,15)<<endl;<>
getch();
}
int volume(int s)
```

```
{
return(s*s*s);
}
double volume(double r,int h)
{
return(3.14*r*r*h);
}
long volume(long l,int b,int h)
{
return(l*b*h);
}
```

Output:

```
1000
80
112500
```

Result:

Thus the implementation of c ++ program for function overloading is executed and the output has been verified.

Ex: No: 1(b) DEFAULT ARGUMENT IN C++

Aim:

To write a c ++ program to implement the default argument

Algorithm:

Step 1: Start the program.

Step 2: Declare the simple interest function with default argument.

Step 3: From main function call the required data.

Step 4: Define simple interest function.

Step 5: Calculating simple interest.

Step 6: Display the details given.

Step 7: Stop the program.

Program:

```
#include

#include

float si(float=1000.00,int=2,float=0.02);

void main()

{

clrscr();

float p,r;

int n;

cout<<"\n enter principle:";

cin>>p;

cout<<"\n enter number of year:";

cin>>n;

cout<<"\n enter rate of interest:";
cin>>r;

cout<<"\n default argument p=1000.00,n=2,&r=0.02";

cout<<"\n simple interest si()="<<si();< span=""></si();<>

cout<<"\n simple interest si(p)="<<si(p);< span=""></si(p);<>

cout<<"\n simple interest si(p,n)="<<si(p,n);< span=""></si(p,n);<>

cout<<"\n simple interest si(p,n,r)="<<si(p,n,r);< span=""></si(p,n,r);<>

getch();

}

float si(float pr,int no,float ra)

{

return((pr*no*ra)/pr);

}
```

Output:

```
Enter Principle: 100

Enter number of year: 2011

Enter rate of interest: 10

Default argument P= 1000.00, n=2 & r=0.02

Simple interest Si () =0.04

Simple interest Si (p) =0.04

Simple interest Si (p, n) =40.219997

Simple interest Si (p, n, r) =20110
```

Result:

Thus the implementation of c ++ program for default argument is executed and the output has been verified.

Ex: No: 2 SIMPLE CLASS DESIGN IN C++, NAME SPACES, OBJECT CREATION

Aim:

To write a c ++ program to implement the simple interest class design, name space, object creation

Algorithm:

Step 1: Start the program.

Step 2: Create the class and declare the variable and functions of the class.

Step 3: In main function, objects are created and the respective functions are called for the Function.

Step 4: Get ‘n’ value and calculation function is repeat by using for loop.

Step 5: Stop the program.

Program:

```
#include

#include

int f1=1,f2=1,f3;

class fib

{

public:

void calculation()

{

f3=f2+f1;

cout<<f3<<endl;< span=""></f3<<endl;<>

f1=f2;

f2=f3;

}

};

void main()
```

```
{
clrscr();
int n;
fib obj;
cout<<"enter the value of n:";
cin>>n;
for(int i=0;i<n;i++)< span=""></n;i++)<>
{
obj.calculation();
}
getch();
}
```

Output:

```
Enter the value of n: 5
2
3
5
8
13
```

Result:

Thus the implementation of c ++ program for simple class design c ++, name space, object creation is executed and the output has been verified.

Ex: No: 3 CLASS DESIGN IN C++ USING DYNAMIC MEMORY ALLOCATION, DESTRUTOR, COPY CONSTRUTOR

Aim:

To write a c ++ program to implement the dynamic memory allocation, destructor, and copy constructor

Algorithm:

Step 1: Start the program.

Step 2: Create class string with two constructors the first is an empty constructor, which allows declaring an array of string, the second constructor initiates the length of the string and allocates unnecessary space for the string to the stored and create the string itself.

Step 3: Then destruct and create a member to concentrate two strings.

Step 4: Destruct the string using the symbol (~)

Step 5: Declare the main function using and display the string S1, S2 and display destructor data and display the concatenate string S3 display, the copy constructed string S4.

Step 6: Stop the program.

Program:

```
#include

#include

#include

int count=0;

class string

{

private:

char*str;

public:

string(char*s)

{

count++;

int len=strlen(s);
str=new char[len+1];
strcpy(str,s);
}
string()
{
delete str;
}
void display()
{
cout<<str<<endl;< span=""></str<<endl;<>
}
void concat(string s1,string s2)
{
int len;
len=strlen(s1.str)+strlen(s2.str);
str=new char[len+1];
strcpy(str,s1.str);
strcat(str," ");
strcat(str,s2.str);
}
~string()
{
cout<<"no of object destroyed"<<cout<<endl;< span=""></cout<<endl;<>
count--;
}
```

```
};
int main()
{
clrscr();
string s1="SUPER";
string s2="STAR";
string s3;
string s4(s1);
s1.display();
s2.display();
s3.concat(s1,s2);
s3.display();
s4.display();
getch();
return(0);
}
```

Output:

```
SUPER
STAR
Number of object destroyed 0x8fce 0390
Number of object destroyed 0x8fce 0390
SUPER STAR
STAR
```

Result:

Thus the implementation of c ++ program for the dynamic memory allocation, destructor and copy constructor is executed and the output has been verified.

Ex: No: 4 OPERAROR OVERLOADING, FRIEND FUNCTION.

Aim:

To write a c ++ program to implement the operator overloading with friend function

Algorithm:

Step 1: Start the program.

Step 2: Create a class complex and declare real and imaginary part value and get the value.

Step 3: Declare the friend function.

Step 4: Declare the operator function for various task.

Step 5: Declare the main function and collect the all data and display it.

Step 6: Stop the program.

Program:

```
#include
#include
class complex
{
private:
float real;
float img;
public:
complex()
{
}
complex(int realpart)
{
real=realpart;
}
void realpart()
{
cout<<"realpart:";
cin>>real;
cout<<"imaginarypart:";
cin>>img;
}
oid outdata()
{
cout<<"("<<real;< span=""></real;<>
cout<<"+i"<<img<<")";< span=""></img<<")";<>
}
friend complex operator+(complex c1,complex c2);
friend complex operator-(complex c1,complex c2);
friend complex operator/(complex c1,complex c2);
friend complex operator*(complex c1,complex c2);
};
complex operator+(complex c1,complex c2)
{
complex c;
c.real=c1.real+c2.real;
c.img=c1.img+c2.img;
return(c);
}
```

```
complex operator-(complex c1,complex c2)
{
complex temp;
temp.real=c1.real-c2.real;
temp.img=c1.img-c2.img;
return(temp);
}
complex operator/(complex c1,complex c2)
{
complex temp;
float qt;
qt=(c2.real*c2.real+c2.img*c2.img);
temp.real=(c1.real*c2.real+c2.img*c2.img)/qt;
temp.img=(c1.img*c2.real-c1.real*c2.img)/qt;
return(temp);
}
complex operator*(complex c1,complex c2)
{
complex temp;
temp.real=(c1.real*c2.real)-(c1.img*c2.img);
temp.img=(c1.real*c2.img)-(c1.img*c2.real);
return(temp);
}
void main()
{
```

```
clrscr();

complex c1,c2,c3;

c1.realpart();

c2.realpart();

c3=c1+c2;

cout<<"addition:";

c3.outdata();

c3=c1-c2;

cout<<endl<<"subtraction:";< span=""></endl<<"subtraction:";<>

c3.outdata();

c3=c1*c2;

cout<<endl<<"multiplication:";< span=""></endl<<"multiplication:";<>

c3.outdata();

c3=c1/c2;

cout<<endl<<"division:";< span=""></endl<<"division:";<>

c3.outdata();

getch();

}
```

Output:

```
Real Part: 1

Imaginary Part: 2

Real Part: 1

Imaginary Part: 2

Addition: (2+i0)

Subtraction: (0+i0)
```

```
Multiplication: (-3+i0)

Division: (1+i0)
```

Result:

Thus the implementation of c ++ program for the operator overloading with friend function is executed and the output has been verified.

Ex: No: 5(a)   OVERLOADING ASSIGNMENT OPERATOR

Aim:

To write a c ++ program to implement the overloading assignment operator

Algorithm:

Step 1: Start the program.

Step 2: Create a different classes.

Step 3: Declare the assignment operator.

Step 4: Declare the operator function for various task.

Step 5: Declare the main function and collect the all data and display it.

Step 6: Stop the program.

Program:

```
#include
#include

using namespace std;

struct A {
  A& operator=(const A&) {
    cout << "A::operator=(const A&)" << endl;
    return *this;
  }

  A& operator=(A&) {
    cout << "A::operator=(A&)" << endl;
    return *this;
  }

};
class B {
  A a;
};

struct C {
  C& operator=(C&) {
    cout << "C::operator=(C&)" << endl;
    return *this;
  }
  C() { }
};

void main()
{
clrscr();

  B x, y;
  x = y;

  A w, z;
  w = z;
  C i;
  const C j();
//  i = j;
getch();
}
```

Output:

```
A::operator= (const a&)
A::operator (A&)
```

Result:

Thus the implementation of c ++ program for overloading assignment operator is executed and the output has been verified.

Ex: No: 5(b) TYPE CONVERSION

Aim:

To write a c ++ program to implement the program for type conversion

Algorithm:

Step 1: Start the program.

Step 2: Declare the main function and get the float value.

Step 3: We can invoke the different type conversion classes.

Step 4: Then display the variables of x1,x2,x3.

Step 5: Stop the program.

Program:

```
#include

#include

void main()

{

clrscr();

float num=98.76;

int x1=(int)num;

int x2=int(num);

int x3=static_cast(num);

cout<<”x1=”<<x1<<endl;< span=""></x1<<endl;<>

cout<<”x2=”<<x2<<endl;< span=""></x2<<endl;<>

cout<<”x3=”<<x3<<endl;< span=""></x3<<endl;<>

getch();

}
```

Output:

```
X1=98
```

```
X2=98

X3=98
```

Result:

Thus the implementation of c ++ program for type conversion is executed and the output has been verified.

Ex: No: 6 INHERITANCES, RUNTIME POLYMORPHISM

Aim:

To write a c ++ program to implement the inheritance and the run-time polymorphism

Algorithm:

Step 1: Start the program.

Step 2: Create a class account with the following member variable name, acc_no, acc_type, and balance.

Step 3: Create derived classes with following member variables

(i).For current account information like balance gets deposit and get withdrawal amount.

(ii).For saving account information like balance get deposit and get withdrawal amount.

Step 4: Write a member function to get the deposit and withdrawal amount and to update the balance information for current and savings etc.

Step 5: Write a member function to display the balance information for respective account.

Step 6: Stop the program.

Program:

```
#include

#include

class account

{

char name[25];

char acc_no[20];

char acc_type[15];

float balance;

public:void readdata()

{

cout<<"enter name:";
cin>>name;
cout<<"account number:";
cin>>acc_no;
cout<<"account type:";
cin>>acc_type;
cout<<"balance:";
cin>>balance;
}
void displayaccountdata()
{
cout<<"name:"<<name<<endl;< span=""></name<<endl;<>
cout<<"account number:"<<acc_no<<endl;< span=""></acc_no<<endl;<>
cout<<"account type:"<<acc_type<<endl;< span=""></acc_type<<endl;<>
cout<<"balance:"<<balance<<endl;< span=""></balance<<endl;<>
}
int bal()
{
return balance;
}
};
class currentaccount:virtual public account
{
protected:
int deposit,withdraw,balance;
public:
```

```
int getdeposit()

{

cout<<"enter the amount to be deposited in the account:"<<endl;< span=""></endl;<>

cin>>deposit;

return deposit;

}

int getwithdraw()

{

cout<<"enter the amount to be withdraw from the account:"<<endl;< span=""></endl;<>

cin>>withdraw;

return withdraw;

}

};

class current:public currentaccount

{

private:

int balance;

public:

void depositbal()

{

balance=getdeposit()+bal();

cout<<"balance amount in the account:"<<balance;< span=""></balance;<>

}

void withdrawbal()

{
```

```
balance=bal()-getwithdraw();

cout<<"balance amont in the account:"<<balance;< span=""></balance;<>

}

};

class savingaccount:virtual public account

{

protected:

int deposit,withdraw,balance;

public:int getdeposit()

{

cout<<"enter the amount to be deposited in the account:"<<endl;< span=""></endl;<>

cin>>deposit;

return deposit;

}

int getwithdraw()

{

cout<<"enter the amount to be deposit in the account:"<<endl;< span=""></endl;<>

cin>>deposit;

return deposit;

}

};

class saving:public savingaccount

{

private:

int balance;
```

```
public:
void depositbal()
{
balance=getdeposit()+bal();
cout<<"balance amount in the account:"<<balance;< span=""></balance;<>
}
void withdraw()
{
cout<<"balance amount in the account:"<<balance;< span=""></balance;<>
}
};
void main()
{
clrscr();
current c1;
int option,choice;
c1.readdata();
cout<<"enter up option:1 current account 2.saving account:"<<endl;< span=""></endl;<>
cin>>option;
if(option==1)
{
cout<<"1.withdraw 2.deposit:"<<endl;< span=""></endl;<>
cin>>choice;
if(choice==2)
{
```

```
c1.displayaccountdata();
c1.depositbal();
}
else if(choice==1)
{
c1.displayaccountdata();
c1.withdrawbal();
}
}
if(option==2)
{
cout<<"1.withdraw 2.deposit:"<<endl;< span=""></endl;<>
cin>>choice;
{
c1.displayaccountdata();
c1.depositbal();
}
if(choice==1)
{
c1.displayaccountdata();
c1.withdrawbal();
}
}
getch();
}
```

Output:

```
Enter Name: VIJAY

Account No: 55555

Account type: AB

Balance: 10000

Enter up option: 1.Current account 2.Savings account

1. Withdraw 2.Deposit

Name: VIJAY

Account No: 55555

Account Type: AB

Balance: 10000

Enter the amount to be deposited in the account: 1000

Balance amount in the account: 11000
```

Result:

Thus the implementation of c ++ program for inheritance, runtime polymorphism is executed and the output has been verified.

Ex: No: 7 I/O THROWING AND CATCHING EXCEPTION

Aim:

To write a Program for Exception Handling Divide by zero Using C++ Programming

Algorithm:

```
Step 1: Start the program.
Step 2: Declare the variables a,b,c.
Step 3: Read the values a,b,c,.
Step 4: Inside the try block check the condition.
a. if(a-b!=0) then calculate the value of d and display.
b. otherwise throw the exception.
Step 5: Catch the exception and display the appropriate message.
Step 6: Stop the program.
```

Program:

```
#include
#include
void main()
{
int a,b,c;
float d;
clrscr();
cout<<"Enter the value of a:";
cin>>a;
cout<<"Enter the value of b:";
cin>>b;
cout<<"Enter the value of c:";
cin>>c;
try
{
if((a-b)!=0)
{
d=c/(a-b);
cout<<"Result is:"<<d;< span=""></d;<>
}
else
{
throw(a-b);
}
}
catch(int i)
{
cout<<"Answer is infinite because a-b is:"<<i;< span=""></i;<>
}
getch();
}
```

Output:

```
Enter the value for a: 20
Enter the value for b: 20
Enter the value for c: 40
Answer is infinite because a-b is: 0
```

Result:

Thus the implementation of c ++ program for exceptional handling is executed and the output has been verified.

Ex: No: 8 TEMPLATE DESIGN IN C ++

Aim:

To write a c ++ program to implement the template design

Algorithm:

Step 1: Start the program.

Step 2: Declare the main function and get the swap variables a, b, and c, d.

Step 3: We can invoke the swap () function like carry ordinary function.

Step 4: Then display the swap variables.

Step 5: Stop the program.

Program:

```
#include

#include

void swap(int,int);

void swap(float,float);

void main()

{

clrscr();

int a,b;

float c,d;

cout<<”ENTER THE VALUES FOR A&B:”<<endl;< span=""></endl;<>

cin>>a>>b;

cout<<”ENTER THE VALUES FOR C&D:”<<endl;< span=""></endl;<>

cin>>c>>d;

swap(a,b);
swap(c,d);

}

template

void swap(t x,t y)

{

t temp;

temp=x;

x=y;

y=temp;

cout<<x<<y;< span=""></x<<y;<>

getch();

}
```

Output:

```
Enter the values for A&B:

13

65

Enter the values for C &D:

23

78

65137823
```

Result:

Thus the implementation of c ++ program for template design is executed and the output has been verified.

Ex: No: 9 PROGRAM DEVELOPMENT USING STL

Aim:

To write the c++ program to implement the program development using STL

Algorithm:

Step 1: Start the program.

Step 2: Declare the main function

Step 3: We can invoke the stack empty(),push(), pop() function like carry ordinary function.

Step 4: Then display the stack variables.

Step 5: Stop the program.

Program:

```
#include

#include "Stack.h"

using namespace std;

int main(){

Stack myStack;

if (myStack.empty())

cout<<"Stack is empty"<<endl;< span=""></endl;<>

else

cout<<"Stack is not empty"<<endl;< span=""></endl;<>

for (int i = 0; i < 10; i++)

myStack.push(i);

cout<<"The top of the Stack is: "<<mystack.top()<<endl;< span=""></mystack.top()<<endl;<>

if (myStack.empty())

cout<<"Stack is empty"<<endl;< span=""></endl;<>

else
cout<<"Stack is not empty"<<endl;< span=""></endl;<>

myStack.writeStack();

cout<<endl<<"my stack="" size="" span=""></endl<<"my>

cout<<endl;< span=""></endl;<>

while (!myStack.empty())

myStack.pop();

if (myStack.empty())

cout<<"Stack is empty"<<endl;< span=""></endl;<>

else

cout<<"Stack is not empty"<<endl;< span=""></endl;<>

cout<<"The top of the Stack is: "<<mystack.top()<<endl;< span=""></mystack.top()<<endl;<>

return 0;

}
```

Output:

```
Stack is empty
the top of the Stack is: 9
Stack is not empty
9 8 7 6 5 4 3 2 1 0
My Stack size is: 10

Stack is empty
top (): Stack is empty
```

Result:

Thus the implementation of c ++ program for development of STL is executed and the output has been verified.

Ex: No: 10(a) SIMPLE CLASS DESIGN IN JAVA

Aim:

To write a JAVA program to implement the sum of n numbers

Algorithm:

```
Step 1: Start the program.
Step 2: Declare n=10 and find the sum of 10 numbers.
Step 3: Initialize s=0.
Step 4: For i value from 1 to 10 repeat the process by using for loop.
Step 5: Display the loop.
Step 6: Stop the program.
```

Program:

```
import java.io.*;
class simple
{
public static void main(String args[])
{
int i, s=0, n=0;
for (i=1, i<=n; i++)
{
s=s+i;
System.out.println("sum of n numbers:"+s);
}
}
}
```

Output:

```
Sum of n number: 1

Sum of n number: 3

Sum of n number: 6

Sum of n number: 10

Sum of n number: 15

Sum of n number: 21

Sum of n number: 28

Sum of n number: 36

Sum of n number: 45

Sum of n number: 55
```

Result:

Thus the implementation of JAVA program for the sum of n numbers is executed and the output has been verified.

Ex: No: 10(b) FIBONACCI SERIES

Aim:

To write a JAVA program to implement the Fibonacci series

Algorithm:

Step 1: Start the program.

Step 2: Declare n=10. So find the Fibonacci series up to 10.

Step 3: Initialize a=-1, b=1.

Step 4: Add a and b value and store it in int.

Step 5: Display the f values.

Step 6: Then assign a=b and b=f.

Step 7: For I from 1 to 10 repeat the step 4, 5 and 6.

Step 8: Stop the program.

Program:

```
import java.io.*;

classfibo

{

public static void main(String args[])

{

int a=-1, b=1, i, f, n=10;

System.out.println("Fibonacci series is\n");

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

{

f=a+b;

System.out.println(f);

a=b;
b=f;
}
}
}
```

Output:

```
0
1
1
2
3
5
8
13
21
34
```

Result:

Thus the implementation of JAVA program for the Fibonacci series is executed and the output has been verified.

Ex: No: 10(c) PRINTING MULTIPLICATION TABLE

Aim:

To write a JAVA program to implement printing the multiplication table.

Algorithm:

```
Step 1: Start the program.
Step 2: Create class multi table and declare the variable column and row.
Step 3: For the row value for 1 to 10 repeat the step 4 and 5.
Step 4: For column value from the 1 to 10 repeat the step 5.
Step 5: Multiply row and column.
Step 6: Print the output data y.
Step 7: Stop the program.
```

Program:

```
import java.io.*;
class multitable
{
public static void main(String args[])
{
int row, col, y;
System.out.println("Multiplication table\n");
for(row=1; row<=10; row++)
{
for(col=1; col<=10; col++)
{
y=row*col;
System.out.print(" "+y);

}

System.out.println("\n");

}

}

}
```

Output:

```
1 2 3 4 5 6 7 8 9 10

2 4 6 8 10 12 14 16 18 20

3 6 9 12 15 18 21 24 27 30

4 8 12 16 20 24 28 32 36 40

5 10 15 20 25 30 35 40 45 50

6 12 18 24 30 36 42 48 54 60

7 14 21 28 35 42 49 56 63 70

8 16 24 32 40 48 56 64 72 80

9 18 27 36 45 54 63 72 81 90

10 20 30 40 50 60 70 80 90 100
```

Result:

Thus the implementation of JAVA program for printing multiplication table is executed and the output has been verified.

Ex: No: 11 DESIGNING PACKAGE WITH JAVA PACKAGE EEE

Aim:

To write a JAVA programs to implement designing package with JAVA package EEE

Algorithms:

Step 1: Start the program.

Step 2: Create a package called name which contains the class addition with member function to perform addition operation.

Step 3: Impact the created package in the main class.

Step 4: From the main class make a function call to the package.

Step 5: Display the resultant values.

Step 6: Stop the program.

Programs:

```
package eee;

import java.io.*;

public class addition

{

int a,b,c;

public void add(int a,int b)

{

c=a+b;

System.out.println("sum is"+c);

}

}

import java.io.*;

//import eee.addition;
```

```
class sum

{

public static void main(String args[])

{

addition d1=new addition();

d1.add(80,20);

}

}
```

Output:

```
Sum is 100
```

Result:

Thus the implementation of JAVA program for Designing package with JAVA package EEE is executed and the output has been verified.

Ex: No: 12 INTERFACES AND INHERITANCE IN JAVA

Aim:

To write a JAVA programs for interface and inheritance.

Algorithm:

```
Step 1: Start the programs.
Step 2: Create and interface named sports which contains member function.
Step 3: Created three classes are inherited.
Step 4: Created an object for the class result.
Step 5: With respect to create object make a function call to the function on other classes including interface.
Step 6: Stop the program.
```

Programs:

```
//Interface program

interface Area

{

float compute(float x, float y);

}

//Inheritance program

class Rectangle implements Area

{

public float compute(float x, float y)

{

return(x * y);

}

}
```

```
class Triangle implements Area
{
public float compute(float x,float y)
{
return(x * y/2);
}
}
class InterfaceArea
{
public static void main(String args[])
{
Rectangle rect = new Rectangle();
Triangle tri = new Triangle();
Area area;
area = rect;
System.out.println("Area Of Rectangle = "+ area.compute(1,2));
area = tri;
System.out.println("Area Of Triangle = "+ area.compute(10,2));
}
}
```

Output:

```
 Area of a rectangle=2.0
Area of a triangle=10.0
```

Result:

Thus the implementation of JAVA program for interface and inheritance is executed and the output has been verified.

Ex: No: 13 EXCEPTIONS HANDLING IN JAVA

Aim:

To write a java program to implement the exception handling.

Algorithm:

Step 1: Start the programs.

Step 2: Include the import statement.

Step 3: Declare the main class method.

Step 4: Declare the class and definite them.

Step 5: In that through a exception in the try.

Step 6: The exception will be caught by the catch block.

Step 7: Stop the program.

Program:

```
import java.io.*;
public class ExceptionTest
{
public static void main(String args[])
{
try
{
int a[]=new int[2];
System.out.println("access element three:"+a[3]);
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("Exception thrown:"+e);
}
System.out.println("out of the block");
}
}
```

Output:

```
 Exception thrown: java.lang.ArrayIndexOutOfBoundsException:3
Out of the block
```

Result:

Thus the implementation of JAVA program for exception handling is executed and the output has been verified.

Ex: No: 14 JAVA I/O

Aim:

To write a JAVA programs to implement the Java I/O program.

Algorithm:

Step 1: Start the program.

Step 2: Declare the main function.

Step 3: create the memory space for indata.

Step 4: Get the data and read the data.

Step 5: Then display the get data.

Step 7: Stop the program

Programs:

```
import java.io.*;

class Echo

{

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

{

InputStreamReader inStream =new InputStreamReader( System.in ) ;

BufferedReader stdin =new BufferedReader( inStream );

String inData;

System.out.println("Enter the data:");

inData = stdin.readLine();

System.out.println("You entered:" + inData );

}

}
```

Output:

```
Enter the data: Hi. . Welcome to E.S

You entered: Hi. . Welcome to E.S
```

Result:

Thus the implementation of JAVA program for Java I/O program is executed and the output has been verified.

Ex: No: 15 DESIGN MULTITHREAD PROGRAM IN JAVA

Aim:

To write a JAVA programs to implement the design multithread program.

Algorithm:

Step 1: Start the program.

Step 2: Create the thread class.

Step 3: Create the main class thread.

Step 4: Create the memory space.

Step 5: Execute the program.

Step 6: Display the output.

Step 7: Stop the program

Program:

```
public class CurrentThreadDemo1

{

public static void main(String args[])

{

Thread t=Thread.currentThread();

System.out.println("current thread:"+t);

t.setName("my thread");

System.out.println("after name change:"+t);

try

{

for(int n=5;n>0;n--)

{

System.out.println(n);

Thread.sleep(1000);

}

}

catch(InterruptedException ie)

{

System.out.println("main thread interrupted");

}

}

}
```

Output:

```
 Current Thread: Thread [Main, 5, Main]
```

```
After Name Change: Thread [My Thread, 5, Main]

5

4

3

2

1
```

Result:

Thus the implementation of JAVA program for design multithread program is executed and the output has been verified.