	Government Girls' Polytechnic, Bilaspur
	Class: **CSE 5th Sem, IT -5th Sem**
	Name of the Lab: **Operating System Lab**
	Title of the Practical: **Java Programming Lab**
	Teachers Assessment: 20 End Semester Examination:50

EXPERIMENT NO:- 1

1.OBJECTIVE :- Programs to design Compound Interest.
2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or Windows xp,Vista.
3.SOFTWARE REQUIRED :- Java.
4.THEORY :- This is the Compound Interest Formula.
a=p*Math.pow(1+r/100,n);
Source Code :

```
class Compound
{
public static void main(String args[])
{
double p,r,n,a,i;
p=100;
r=7;
n=4;
a=p*Math.pow(1+r/100,n);
i=a-p;
System.out.println(i);
}
}
```

5.PROGRAM INPUTS & OUTPUT :-
31.079601000000025
6.OBSERVATIONS :- Task is Performed.

EXPERIMENT NO:- 2

1.OBJECTIVE :- Program to design comparison two Variable use if statements.
2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or Windows xp,Vista.
3.SOFTWARE REQUIRED :- Java.
4.THEORY :-
Here comparison two Variable use if statements.
Source Code :

```
class IfSample{
public static void main(String args[])
{
int x,y;
x=10;
y=20;
if(x<y) System.out.println("x is less then y");
x=x*2;
if(x==y)System.out.println("x now equal to y");
x=x*2;
if(x>y)System.out.println("x now greater then y");
if(x==y)System.out.println("your won't see this");
}
}
```

5.PROGRAM INPUTS & OUTPUT :-
x is less then y
6.OBSERVATIONS :- Task is Performed.

EXPERIMENT NO:- 3

1.OBJECTIVE :- Program to design Constructor.
2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or Windows xp,Vista.
3.SOFTWARE REQUIRED :- Java.
4.THEORY :-
Constructor is a special member function of class call automatically when object is created.
Source Code :

```
class Rectangle
{
int length,width;
Rectangle(int x,int y)
{
length = x;
width = y;
}
int rectArea()
{
return(length * width);
}
}
class Constructors
{
public static void main(String args[])
{
Rectangle rect1 = new Rectangle(15,10);
int area1 = rect1.rectArea();
System.out.print("Area1 = "+area1);
}
}
```

5.PROGRAM INPUTS & OUTPUT :-
Area1 = 150
6.OBSERVATIONS :- Task is Performed.

EXPERIMENT NO:- 4

1.OBJECTIVE :- Program to design toUpperCase() and toLowerCase Type Conversion.
2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or Windows xp,Vista.
3.SOFTWARE REQUIRED :- Java.
4.THEORY :-
The method **toLowerCase( )** converts all the characters in a string from uppercase to lowercase. The **toUpperCase( )** method converts all the characters in a string from lowercase to uppercase. Nonalphabetical characters, such as digits, are unaffected.
Here are the general forms of these methods:

String toLowerCase( )
String toUpperCase( )

Source Code :

```
class ChangeCase {
public static void main(String args[])
{
String s = "This is a test.";
System.out.println("Original: " + s);
String upper = s.toUpperCase();
String lower = s.toLowerCase();
System.out.println("Uppercase: " + upper);
System.out.println("Lowercase: " + lower);
}
}
```

5.PROGRAM INPUTS & OUTPUT :-
Original: This is a test.
Uppercase: THIS IS A TEST.
Lowercase: this is a test.
6.OBSERVATIONS :- Task is Performed.

EXPERIMENT NO:- 5

1.OBJECTIVE :- Programs to design Single Inheritance.
2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or Windows xp,Vista.
3.SOFTWARE REQUIRED :- Java.
4.THEORY :-
Inheritance is the process by which object of one class acquire the properties of object of another class.

Source Code :

```
class Room
{
int length;
int breadth;
Room(int x,int y)
{
length = x;
breadth = y;
}
int area()
{
return(length * breadth);
}
}
class BedRoom extends Room
{
int height;
BedRoom(int x,int y,int z)
{
super(x,y);
height = z;
}
int volume()
{
return(length * breadth * height);
}
}
class Inheritance
{
public static void main(String args[])
{
BedRoom room1 = new BedRoom(14,12,10);
int area1 = room1.area();
int volume1 = room1.volume();
System.out.println("Area1 = "+ area1 );
System.out.println("Volume1 = "+ volume1);
}
}
```

5.PROGRAM INPUTS & OUTPUT :-
Area1 = 168
Volume1 = 1680
6.OBSERVATIONS :- Task is Performed.

EXPERIMENT NO:- 6

1.OBJECTIVE :- Programs to design interface.
2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or Windows xp,Vista.
3.SOFTWARE REQUIRED :- Java.
4.THEORY :-
Using the keyword interface ,you can fully abstract a class' interface from its implementation.
Source Code :

```
interface Area
{
final static float pi=3.14F;
float compute(float x,float y);
}
class Rectangle implements Area
{
public float compute(float x,float y)
{
return(x*y);
}
}
class Circle implements Area
{
public float compute(float x,float y)
{
return(pi*x*x);
}
}
class Interface
{
public static void main(String args[])
{
Rectangle rect=new Rectangle();
Circle cir=new Circle();
Area area;
area =rect;
System.out.println("Area of Rectangle="+area.compute(10,20));
area =cir;
System.out.println("Area of Circle="+area.compute(10,0));
}
}
```

5.PROGRAM INPUTS & OUTPUT :-
Area of Rectangle=200.0
Area of Circle=314.0
6.OBSERVATIONS :- Task is Performed.

EXPERIMENT NO:- 7

1.OBJECTIVE :- Program to design exception by use Try-Catch-Finally.
2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or Windows xp,Vista.
3.SOFTWARE REQUIRED :- Java.
4.THEORY :-
Java exception handling is managed via five keywords: **try**, **catch**, **throw**, **throws**, and **finally**.
Source Code :

```
class FinallyDemo {
// Through an exception out of the method.
static void procA() {
try {
System.out.println("inside procA");
throw new RuntimeException("demo");
} finally {
System.out.println("procA's finally");
}
}
public static void main(String args[]) {
try {
procA();
} catch (Exception e) {
System.out.println("Exception caught");
}
}
}
```

5.PROGRAM INPUTS & OUTPUT :-
From Thread C : k = 5
Exit from C
6.OBSERVATIONS :- Task is Performed.

EXPERIMENT NO:- 8

1.OBJECTIVE :- Program to design multi-threading.
2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or Windows xp,Vista.
3.SOFTWARE REQUIRED :- Java.
4.THEORY :-
A multithreaded program contains two or more parts that can run concurrently.Each part of such a program is called a thread , and each thread define a separate path of execution.
Source Code :

```
class A extends Thread
{
public void run()
{
for(int i=1;i<=5;i++)
{
System.out.println("From Thread A : i = " +i);
}
System.out.println("Exit from A");
}
}
class B extends Thread
{
public void run()
{
for(int j=1;j<=5;j++)
{
System.out.println("From Thread B : j = " +j);
}
System.out.println("Exit from B ");
}
}
class C extends Thread
{
public void run()
{
for(int k=1;k<=5;k++)
{
System.out.println("From Thread C : k = " +k);
}
System.out.println("Exit from C ");
}
}
class ThreadTest
{
public static void main(String args[])
{
new A().start();
new B().start();
new C().start();
}
}
```

5.PROGRAM INPUTS & OUTPUT :-

```
From Thread C : k = 4
```

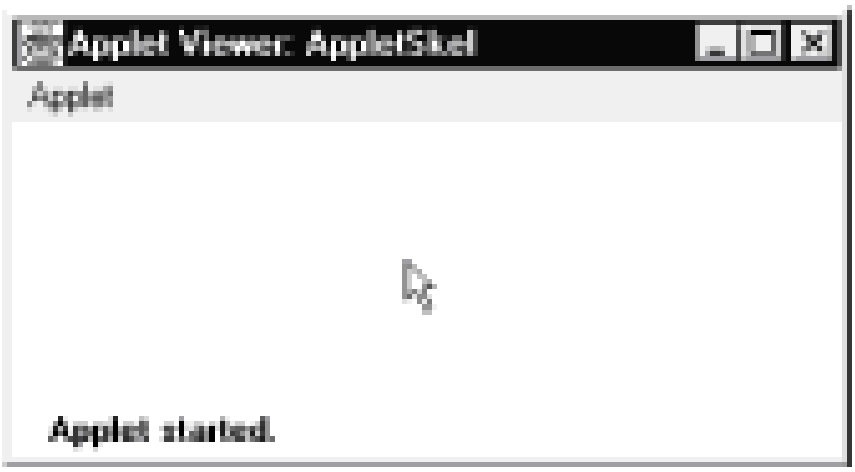
From Thread C : k = 5
Exit from C
6.OBSERVATIONS :- Task is Performed.

EXPERIMENT NO:- 9

1.OBJECTIVE :- Program to design Applet skeleton.
2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or Windows xp,Vista.
3.SOFTWARE REQUIRED :- Java.
4.THEORY :-
the most trivial applets override a set of methods that provides the basic mechanism by which the browser or applet viewer interfaces to the applet and controls its execution.
Source Code :

```
import java.awt.*;
import java.applet.*;
/*
<applet code="AppletSkel" width=300 height=100>
</applet>
*/
public class AppletSkel extends Applet {
// Called first.
public void init() {
// initialization
}
/* Called second, after init(). Also called whenever
the applet is restarted. */
public void start() {
// start or resume execution
}
// Called when the applet is stopped.
public void stop() {
// suspends execution
}
/* Called when applet is terminated. This is the last
method executed. */
public void destroy() {
// perform shutdown activities
}
// Called when an applet's window must be restored.
public void paint(Graphics g) {
// redisplay contents of window
}
}
```

5.PROGRAM INPUTS & OUTPUT :-



6.OBSERVATIONS :- Task is Performed.