1 a) Develop Java Program to demonstrate Constructor Overloading and Method Overloading.

```
package Experiment;
class Student{
	String name;
	int regno;
	int marks1,marks2,marks3;
	//	null or default constructor
	Student(){
		name="Abdus Salam";
		regno=12345;
		marks1=90;
		marks2=93;
		marks3=96;
	}
	//	parameterized constructor
	Student(String n,int r,int m1,int m2,int m3)
	{
		name=n;
		regno=r;
		marks1=m1;
		marks2=m2;
		marks3=m3;

	}
	//	copy constructor
	Student(Student s)
	{
		name=s.name;
		regno=s.regno;
		marks1=s.marks1;
```

```
            marks2=s.marks2;
            marks3=s.marks3;
        }
        void display()
        {
            System.out.println(name+"\t"+regno+"\t"+marks1+"\t"+marks2+"\t"+marks3);
        }
        void display(Student s)
        {
            System.out.println("Method Overloading");
        System.out.println(s.name+"\t"+s.regno+"\t"+s.marks1+"\t"+s.marks2+"\t"+s.mark3);
        }
}
public class LabPgrm1 {
        public static void main(String[] args) {
            // TODO Auto-generated method stub
            Student s1=new Student();
            Student s2=new Student("Armaan Malik",342566,98,95,93);
            Student s3=new Student(s2);
            s1.display();
            s2.display(s2);
            s3.display();
        }
}
```

Output:

```
Abdus Salam   12345  90       93       96
Method Overloading
Armaan Malik 342566          98       95       93
Armaan Malik 342566          98       95       93
```

1 b)Develop Java Program to implement Inner class and demonstrate its Access protection.

```
package Experiment;

//public class LabPgrm1B {
class Outer {
	int m = 10;
	void outerdisplay() {
		System.out.println("The base class value of m is " + m);
	}
	class Inner {
		int n = 20;
		void innerdisplay() {
			System.out.println("m = " + m);
			System.out.println("n = " + n);
			outerdisplay(); // call the disp() method of the Outer class using
Outer.this
		}
	}
}
public class LabPgrm1B {
	public static void main(String[] args) {
		Outer outobj = new Outer();
		outobj.outerdisplay();
		Outer.Inner inobj = outobj.new Inner();
		inobj.innerdisplay();
	}
}
```

Output:

```
The base class value of m is 10
m = 10
n = 20
The base class value of m is 10
```

2) Develop Java program in Java for String handling which performs the following:
i) Checks the capacity of StringBuffer objects.

```
package Experiment;
import java.util.Scanner;
public class Lab2 {

	public static void main(String[]args) {
		Scanner in=new Scanner(System.in);
		StringBuffer sb=new StringBuffer();
```

```
                StringBuffer sb1=new StringBuffer("SIT");
                StringBuffer sb2=new StringBuffer(100);

                System.out.println("capacity of Stringbuffer object"+sb.capacity());
                System.out.println("capacity of Stringbuffer object"+sb1.capacity());
                System.out.println("capacity of Stringbuffer object"+sb2.capacity());
                System.out.println("Enter string name to reverse");

                String s=in.nextLine();
                String r=new String();

                for(int i=s.length() -1;i>=0;i--)
                        r+=s.charAt(i);
                        System.out.println("Reverse of " + s + " is " +r);
                        r=r.toUpperCase();
                        System.out.println("its Uppercase " +r);
                   System.out.println("enter sTRING to append");
                        String a = in.nextLine();
                        System.out.println("appending string " +a+ " with "+" is " +
a.concat(r));
                        in.close();



        }
}
```

Output:

capacity of Stringbuffer object16

capacity of Stringbuffer object19

capacity of Stringbuffer object100

Enter string name to reverse

salman

Reverse of salman is namlas

its Uppercase NAMLAS

enter sTRING to append

salman

appending string salman with is salmanNAMLAS

ii)demonstrate Usage of this keyword:

```
package Experiment;
import java.util.Scanner;
class ThisExample
{
	String name;
	int rollno;
	public ThisExample()
	{
		this("SIT",1);
		System.out.println("Default constructor");
	}
	ThisExample(String name,int rollno)
	{
		this.name=name;
		this.rollno=rollno;
		System.out.println("parameter constructor");
	}
	void getdata()
	{
		Scanner sc=new Scanner(System.in);
	   System.out.println("Enter your name");
		name=sc.nextLine();
		System.out.println("Enter your Rollnumber");
		rollno=sc.nextInt();
		sc.close();
	}
```

```
	void display()
	{
		getdata();
		System.out.println("Name is "+ name +" Roll number is "+rollno);
	}
}
public class ThisDemo
{
	public static void main(String[] args)
	{
		ThisExample ex=new ThisExample();
		ex.display();

	}
}
```

3 a) Develop Java Program to demonstrate Inheritance.

```
package Experiment;
import java.util.Scanner;
class HCalculator {
	double num1;
	double num2;
	HCalculator()
	{
		num1=num2=0;
	}
	HCalculator(double n1,double n2)
	{
		num1=n1;
		num2=n2;
	}
	void add()
	{
		System.out.println("The sum is"+(num1+num2));
		sub();
	}
	void sub()
	{
		System.out.println("The Diff is"+(num1-num2));
		mul();
	}
	void mul() {
		System.out.println("The Mul is"+(num1*num2));
		div();
```

```
	}
	void div()
	{
		if(num2!=0)
			System.out.println("The div is"+(num1/num2));
		else
			System.out.println("The Num2 value is zero");
	}
}
public class HLab3A extends HCalculator{

	void accept()
	{
		Scanner read=new Scanner(System.in);
		System.out.println("Enter First Number");
		num1=read.nextDouble();
		System.out.println("Enter Second Number");
		num2=read.nextDouble();
		read.close();
		add();

	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		HLab3A a=new HLab3A();
		a.accept();

	}

}
```

3b) Develop Java program for the implementation of Multiple inheritance using interfaces
to calculate the area of a rectangle and triangle.

```
package Experiment;

class Areacalc implements Rectangle,Triangle123

{

	@Override

	public double Tarea(double x, double y) {

		// TODO Auto-generated method stub


		return(0.5*x*y);

	}

	@Override

	public double Rarea(double x, double y) {

		// TODO Auto-generated method stub


		return(x*y);

	}


}

public class Lab3b {


	public static void main(String[] args) {

		// TODO Auto-generated method stub

		Areacalc a=new Areacalc();

		System.out.println("\n Area of Rectangle is :"+a.Rarea(2, 5));

		System.out.println("\n Area of Triangle is :"+a.Tarea(2, 5));
```

```
	}


}
```

4) Develop 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.

```
package Experiment;

class Account{
	int bal;
	public Account(int bal)
	{
		this.bal=bal;


	}
	void deposit(int amt) {
		bal+=amt;
		System.out.println("current bal="+bal);
	}
	void withdraw(int amt)throws LessBalanceException{
		int need=bal-amt;
		if(need<500)
			throw new LessBalanceException(amt);
		else
			bal-=amt;
		System.out.println("current bal:"+bal);
```

```
	}
}
class LessBalanceException extends Exception{
	int need;
	public LessBalanceException(int need) {
		this.need=need;
	}
	public String toString() { //overriding tostring method of object
		return "withdraw amt Rs  "+need+" " + " is not valid ";
	}
}
public class Lab4 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Account a1=new Account(500);
		Account a2=new Account(500);
		a1.deposit(1000);
		a2.deposit(1000);
		try {
			a1.withdraw(900);
			a2.withdraw(1200);
		}
		catch(LessBalanceException lbe) {
			System.out.println(lbe);
		}
	}

}
```

5) Develop Java program using Synchronized Threads, which demonstrates Producer Consumer concept.

```
class Q {
    int n;
    boolean valueset=false;
    synchronized int get(){
        while(!valueset){
            try {
                wait(10000);
            }catch(InterruptedException e){
                System.out.println("InterruptedException caught");
            }
        }
        System.out.println("Got:"+n);
        valueset=false;
        notify();
        return n;
    }
    synchronized void put(int n){
        while(valueset){
            try {
                wait(10000);
            }catch(InterruptedException e){
                System.out.println("InterruptedException caught");
            }
        }
        this.n=n;
        valueset=true;
        System.out.println("put"+n);
        notify();
```

```
    }}
class Producer implements Runnable {
    Q q;
    Producer(Q q){
        this.q=q;
        new Thread(this,"producer").start();
    }
    public void run(){
        int i=0;
        while(true){
            q.put(i++);
        }
    }
}
class Consumer implements Runnable {
    Q q;
    Consumer(Q q){
        this.q=q;
        new Thread(this,"consumer").start();
    }
    public void run(){
        while(true){
            q.get();
        }
    }
}
public class Lab5 {
            public static void main(String[] args) {
        Q q=new Q();
        new Producer(q);
```

```
        new Consumer(q);
        System.out.println("Press Control-c to stop.");
          }
}
```

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

```
import java.util.*;

class QueueError extends Exception {
        public QueueError(String msg) {
                super(msg);
        }
}

class Que {
        private int size;
        private int front = -1;
        private int rear = -1;
        private int[] queArr;

        public Que(int size) {
                this.size = size;
                queArr = new int[size];
        }

        public void insert(int item) throws Exception, QueueError {
                try {
                        if (rear == size - 1) {
                                throw new QueueError("Queue Overflow");
                        } else if (front == -1) {
                                rear++;
                                queArr[rear] = item;
                                front = rear;
                        } else {
                                rear++;
                                queArr[rear] = item;
                        }
                } catch (QueueError qe) {
                        qe.printStackTrace();
                }
        }

        public void delete() throws Exception,QueueError
{
try
```

```
{
if(front == -1)
{
throw new QueueError("Queue Underflow");
}
else if(front==rear)
{
System.out.println("\nRemoved "+queArr[front]+" from Queue");
queArr[front] = -1;
front=rear=-1;
}
else
{
System.out.println("\nRemoved "+queArr[front]+" from Queue");
front++;
}
}
catch(QueueError qe)
{
qe.printStackTrace();
}
}
```

```
	public void display() throws Exception, QueueError {
		try {
			if (front == -1)
				throw new QueueError("Queue is Empty");
			else {
				System.out.print("\nQueue is: ");
				for (int i = front; i <= rear; i++) {
					System.out.print(queArr[i] + "\t");
				}
				System.out.println();
			}
		} catch (QueueError qe) {
			qe.printStackTrace();
		}
	}
}

class Lab6 {
	public static void main(String[] args) throws Exception,QueueError
{
		int ele;
		System.out.println("\n\n\tQueue test using Array\n\n");
Scanner scan = new Scanner(System.in);
System.out.print("Enter size of Queue array: ");
int size = scan.nextInt();
Que que = new Que(size);
try
```

```
{
while(true)
{
System.out.println("\n\n\tQueue operations \n");
System.out.println("1. Insert");
System.out.println("2. Delete");
System.out.println("3. Display");
System.out.println("4. Exit\n");
System.out.print("Enter your choice: ");
int choice = scan.nextInt();
switch(choice)
{
case 1: System.out.print("\nEnter integer number to insert:");
            ele=scan.nextInt();
 que.insert(ele);
 break;
case 2:que.delete();
 break;
case 3:que.display();
 break;
case 4:exit(1) ;
}
}
}
catch(QueueError qe)
{
            System.out.println("In main we caught Error");
            qe.printStackTrace();
}
}

            private static void exit(int i) {

            }
}
```

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.

```
package Demo;
import Shape.Shapes;
public class Lab7 {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
  Shapes obj=new Shapes();
  Shapes.Square s=obj.new Square(5);
  Shapes.Triangle t=obj.new Triangle(1,1);
  Shapes.Circle c= obj.new Circle(1);
  s.findSquare();
  t.findTriangle();
  c.findCircle();
	}
}

package Shape;
public class Shapes {
	public class Square{
		double sq;
		public Square(double s) {
			sq=s;
		}
			public void findSquare() {
				System.out.println("Square of a number "+sq+"are:"+(sq*sq));

			}
		}
		public class Triangle{
			double l,b;
			public Triangle(double a,double b) {
```

```
                l=a;
                this.b=b;
                }
        public void findTriangle() {
                System.out.println("Triangle: "+(0.5*l*b));
        }
    }
public class Circle{
    double r;
    public  Circle(double a) {
        r=a;
    }
    public void findCircle() {
        System.out.println("circle="+(r*r*3.142));
    }
}
}
```

8) Develop Java Program to create an enumeration Day of Week with seven values SUNDAY through SATURDAY. Add a method isWorkday( ) 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.

CREATE ENUM CLASS:

```
public enum DayOfWeek {
	MONDAY(1),TUESDAY(2),WEDNESDAY(3),THURSDAY(4),FRIDAY(5),SATURDAY(6),SUNDAY(7);
	private int val;
	DayOfWeek(int val)
	{
		this.val=val;
	}
	boolean isworkday()
	{
		if(val<6)
			return true;
		else
			return false;
	}
}
```

CREATE MAIN CLASS:

```
import java.util.Scanner;

public class Lab8 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("1.Monday\n2.Tuesday\n");
		System.out.println("3.Wednesday\n4.Thursday\n");
		System.out.println("5.Friday\n6.Saturday\n");
		System.out.println("7.Sunday\n8.Exit");
		while(true)
		{
			System.out.println("\nSelect your choice");
			Scanner scan=new Scanner(System.in);
			int n=scan.nextInt();
			switch(n)
			{
			case 1:System.out.println("Monday is workingday:"+DayOfWeek.MONDAY.isworkday());
				break;
			case 2:System.out.println("Tuesday is workingday:"+DayOfWeek.TUESDAY.isworkday());
				break;
```

```
                        case 3:System.out.println("Wednesday is
workingday:"+DayOfWeek.WEDNESDAY.isworkday());
                                break;
                        case 4:System.out.println("Thursday is
workingday:"+DayOfWeek.THURSDAY.isworkday());
                                break;
                        case 5:System.out.println("Friday is
workingday:"+DayOfWeek.FRIDAY.isworkday());
                                break;
                        case 6:System.out.println("Saturday is
workingday:"+DayOfWeek.SATURDAY.isworkday());
                                break;
                        case 7:System.out.println("Sunday is
workingday:"+DayOfWeek.SUNDAY.isworkday());
                                break;
                        case 8:System.exit(0);
                        default:System.out.println("Invalid number");
                        }
                }

        }

}
```