Practical 1:

Write a Programme to make a simple Calculator

```
import java.util.*;


public class calculator {

    public static void main(String args[]){

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number : ");

        double a = sc.nextInt();
        double b = sc.nextInt();

        System.out.print("Enter the operator : ");
        char c = sc.next().charAt(0);

        switch(c){

            case '+':
            System.out.println("The addition : " + (a+b));
            break;


            case '-':
            System.out.println("The Subtraction : " + (a-b));
            break;

            case '*':
            System.out.println("The Multiplication : " + (a*b));
            break;

            case '/':
            System.out.println("The Division : " + (a/b));
            break;
        }

        sc.close();

    }
}
```

OUTPUT:

```
Enter the number : 4
5
Enter the operator : +
The addition : 9.0
```

Practical 2:

Write a programme to check a number is palindrome or not

```
import java.util.Scanner;

class Palindrome{
   public static void main (String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter number");
      int num = sc.nextInt();
      int temp=num;
      int rev=0;
      while(temp!=0){
         int rem=temp%10;
         rev=rev*10+rem;
         temp=temp/10;
      }
      System.out.println(rev);
      if(num==rev){
      System.out.println("number is plaindrome");
      }
      else {
         System.out.println("number is not a plaindrome");
      }
      sc.close();
   }
}
```

OUTPUT:

```
Enter number
442
244
number is not a plaindrome
```

Practical 3:

Write a programme to check a number is prime or not between given range

```
import java.util.*;

class prime {

    public static void main(String[] args) {
        System.out.println("Enter the number : ");

        Scanner sc = new Scanner(System.in);

        int a = sc.nextInt();
        int count = 1;

        for (int i = 2; i < a/2; i++) {

            if (a % i == 0) {
                count = 0;
                break;
            } else {
                count = 1;
            }
        }

        if (count == 0) {
            System.out.println(a + " is not a prime number");
        } else {
            System.out.println(a + " is a prime number");
        }
```

```
        sc.close();
    }
}
```

OUTPUT:

```
Enter the number :
2
2 is a prime number
```

Practical 4:

Write a programme to implement matrix multiplication

```
import java.util.*;

public class MatrixMulti {

    static void multi(int a[][], int b[][], int res[][], int N) {

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                for (int k = 0; k < N; k++) {
                    res[i][j] += a[i][k] * b[k][j];
                    // System.out.println(a[i][j]+" a ");
                }
            }
        }

        System.out.println("The matrix multiplication is : ");

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {

                System.out.print(res[i][j]);
                System.out.print(" ");
            }
            System.out.println("");
        }
    }

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of matrix = ");
        int N = sc.nextInt();
        int a[][] = new int[N][N];
        int b[][] = new int[N][N];
        int res[][] = new int[N][N];

        System.out.println("Enter the elements for first martix : ");
```

```
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                a[i][j] = sc.nextInt();
            }
        }

        System.out.println("Enter the elements for Second martix : ");
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                b[i][j] = sc.nextInt();
            }
        }

        multi(a, b, res, N);

        sc.close();
    }
}
```

OUTPUT:

```
Enter the size of matrix = 3
Enter the elements for first martix :
4
5
6
2
1
4
5
6
7
Enter the elements for Second martix :
3
2
5
6
7
2
1
4
5
The matrix multiplication is :
48 67 60
16 27 32
58 80 72
```

Practical 5:

Write a programme to implement sum of digits of a number

```
import java.util.*;

class SumOfDigits {

    public static void main(String args[]) {

        System.out.print("Enter the number = ");
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int temp = n;
        int r, sum = 0;

        while (temp != 0) {

            r = temp % 10;
            sum += r;
            temp = temp / 10;

        }

        System.out.println(sum);
        sc.close();
    }
}
```

OUTPUT:

```
Enter the number = 123
6
```

Practical 6:

Write a programme to implement a number is Armstrong or not

```
import java.util.*;

class ArmStrong {

    public static void main(String args[]){

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number = ");
        int n = sc.nextInt();
        int c=n;
        int temp = n;
        double ans = 0;
        int p=0;

        while(c!=0){
            ++p;
            c=c/10;
        }
        // System.out.println(p);

        while(n!=0){
            int r = n%10;

            ans = ans + Math.pow(r, p);

            n=n/10;
        }

        if(ans == temp){
            System.out.print("It is a Armstrong number ");
        }
        else{
            System.out.print("Not a Armstrong number ");
        }

        sc.close();
```

```
  }

}
```

OUTPUT:

```
Enter the number = 153
It is a Armstrong number
```

Practical 7:

Write a programme to implement dynamic stack

```
import java.util.Scanner;

class DStack{
    int capacity = 2;
    int stack[] = new int[capacity];
    int top = 0;

    void push(int data){
        if(size()==capacity)
            expand();
        stack[top]=data;
        top++;
    }
    void expand(){
        int newStack[] = new int[capacity*2];
        for(int i=0;i<capacity;i++){
            newStack[i]=stack[i];
        }
        stack = newStack;
        capacity*=2;
    }
    int pop(){
        int value;
        top--;
        value = stack[top];
        stack[top]=0;
        return value;
    }
    int peek(){
        return stack[top-1];
    }
    int size(){
        return top;
    }
    boolean empty(){
        return top<=0;
    }
```

```
    void show(){
        for(int n:stack){
            System.out.println(n );
        }
    }
}
class Stack{
    public static void main (String[] args) {
        DStack s = new DStack();
        s.push(15);
        s.push(20);
        s.push(30);
        s.push(15);
        s.show();
        System.out.println("pop");
        s.pop();
        s.show();
        System.out.println("after pop");
        s.push(24);
        s.show();
    }
}
```

OUTPUT:

```
15
20
30
15
pop
15
20
30
0
after pop
15
20
30
24
```

Practical 8:

Write a programe to demonstrate constructor overloading and method overloading

```
class ConstOver {

    int a, b, c;

    ConstOver() {
        a = b = c = 1;
    }

    ConstOver(int d_a, int d_b, int d_c) {

        a = d_a;
        b = d_b;
        c = d_c;

    }

    int volume() {
        return a * b * c;
    }

    void printData() {

        System.out.println("The deafult data = 0 ");
    }

    void printData(int a, int b, int c) {

        System.out.println("Int method :");

        System.out.print("A = " + a);
        System.out.print(" B = " + b);
        System.out.println(" C = " + c);
    }

    void printData(double a, double b, double c) {
```

```
        System.out.println("Double method :");
        System.out.print("A = " + a);
        System.out.print(" B = " + b);
        System.out.println(" C = " + c);
    }

    public static void main(String args[]) {

        ConstOver obj1 = new ConstOver();
        ConstOver obj2 = new ConstOver(4, 5, 6);
        ConstOver obj3 = new ConstOver();

          System.out.println("");

        System.out.println("Constructor Overloading : ");
        int vol = obj1.volume();
        System.out.println("the volume is (default constructor) = " + vol);
        vol = obj2.volume();
        System.out.println("the volume is (Parameterised constructor) = " + vol);

        System.out.println("");

        System.out.println("The method Overloading : ");
        obj1.printData();
        obj2.printData(10, 20, 30);
        obj3.printData(1.5, 2.8, 3.9);

    }

}
```

OUTPUT:

```
Constructor Overloading :
the volume is (default constructor) = 1
the volume is (Parameterised constructor) = 120

The method Overloading :
The deafult data = 0
Int method :
A = 10 B = 20 C = 30
Double method :
A = 1.5 B = 2.8 C = 3.9
```

Practical 9:

Write a programme to set up an array of 10 variables each containing an arbitrary string of form month date year for example 30/10/19 and output as 30th October 1999

```
import java.util.*;

public class Main {

   public static void main(String args[]){

      Scanner sc = new Scanner(System.in);

      System.out.print("Enter the data as DD/MM/YYYY :");

     String data = sc.nextLine();
     data = data.replaceAll("\\s", "");
     String date = data.substring(0,1);
     String month = data.substring(2, 3);
     String year = data.substring(3,7);
     String mon;

      switch(month){

          case "1" :
          mon = "Jan";
          break;

          case "2" :
          mon = "Feb";
          break;

          case "3" :
          mon = "March";
          break;

          case "4" :
          mon = "April";
          break;
```

```
        case "5" :
        mon = "May";
        break;

        case "6" :
        mon = "June";
        break;

        case "7" :
        mon = "July";
        break;

        case "8" :
        mon = "Augut";
        break;

        case "9" :
        mon = "Spet";
        break;

        case "10" :
        mon = "Oct";
        break;

        case "11" :
        mon = "Nov";
        break;

        case "12" :
        mon = "Dec";
        break;

        default :
        mon = "not found ";
    }

  System.out.print(date+"th ");
  System.out.print(mon + " ");
  System.out.print(year);
```

```
        sc.close();

    }
}
```

OUTPUT:

```
Enter the data as DD/MM/YYYY :12 1 2003
12th Jan 2003
```

Practical 10 :

```
import java.util.Scanner;

class Obj{
    int m,cm,mm;
    Obj(int m,int cm, int mm) {
    this.m=m;
    this.cm=cm;
    this.mm=mm;
    }
    void add(Obj o1,Obj o2) {
        int mm2,mm1;
        mm1=1000*o1.m+10*o1.cm+o1.mm;
        mm2=1000*o2.m+10*o2.cm+o2.mm;
        mm1=mm1+mm2;
        int mm3=mm1%10;
        mm1=mm1/10;
        int mm4=mm1%100;
        int mm5=mm1/100;
        System.out.println("Addition is :"+mm5+"m "+mm4+"cm "+mm3+"mm");
    }
    void subtract(Obj o1,Obj o2) {
        int mm2,mm1;
        mm1=1000*o1.m+10*o1.cm+o1.mm;
        mm2=1000*o2.m+10*o2.cm+o2.mm;
        mm1=mm1-mm2;
        int mm3=mm1%10;
        mm1=mm1/10;
        int mm4=mm1%100;
        int mm5=mm1/100;
        System.out.println("Subtraction is :"+mm5+"m "+mm4+"cm "+mm3+"mm");
    }
    void area(Obj o1,Obj o2) {
        int mm2,mm1;
        mm1=1000*o1.m+10*o1.cm+o1.mm;
        mm2=1000*o2.m+10*o2.cm+o2.mm;
        mm1=mm1*mm2;
        int mm3=mm1%100;
        mm1=mm1/100;
        int mm4=mm1%10000;
        int mm5=mm1/10000;
        System.out.println("Area is :"+mm5+"m^2 "+mm4+"cm^2 "+mm3+"mm^2");
}
```

```
}
class prac10{
   public static void main(String args[]) {

       Scanner sc = new Scanner(System.in);
        System.out.println("Enter numbers for second object:");
       int num1 = sc.nextInt();
       int num2 = sc.nextInt();
       int num3 = sc.nextInt();

       System.out.println("Enter numbers for second object:");
       int num4 = sc.nextInt();
       int num5 = sc.nextInt();
       int num6 = sc.nextInt();

       System.out.println("Values are :");
       System.out.println(num1 + " m " + num2 + " cm " + num3 + " mm ");
       System.out.println(num4 + " m " + num5 + " cm " + num6 + " mm ");

       Obj o1=new Obj(num1,num2,num3);
       Obj o2=new Obj(num4,num5,num6);
       o1.add(o1,o2);
       o1.subtract(o2,o1);
       o1.area(o1,o2);

       sc.close();
   }
}
```

OUTPUT:

```
10 }
Enter numbers for second object:
250 10 150
Enter numbers for second object:
32 350 120
Values are :
250 m 10 cm 150 mm
32 m 350 cm 120 mm
Addition is :285m 87cm 0mm
Subtraction is :-214m -63cm 0mm
Area is :323m^2 9704cm^2 8mm^2
```

Practical 11:
Write a programme to implement factorial of a number using recursion

```
import java.util.*;

public class Main {

    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int ans = fact(sc.nextInt());
        System.out.println(ans);

    }

    public static int fact(int n) {
        if(n >= 1){
            return n*fact(n-1);
        }else{
            return 1;
        }
    }

}
```

OUTPUT:

```
5
120
```

Practical 12:
Write a programme to implement GCD of a number using recursion

```
import java.util.*;
public class gcd {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int ans = gcd(sc.nextInt(),sc.nextInt());
        System.out.println(ans);


    }

    public static int gcd(int a, int b) {

        if (b == 0)
            return a;
        else {

            return gcd(b, (a % b));
        }
    }
}
```

OUTPUT:

```
4 6
2
```

Practical 13:

Write a programme to find out that if given matrix is magic square matrix or not

```
import java.util.*;

public class Main {

    static boolean magic(int a[][], int N) {

        int RowSum = 0;
        int ColSum = 0;
        int DiaSum1 = 0;
        int DiaSum2 = 0;

        for (int i = 0; i < N; i++) {
            RowSum = 0;
            ColSum = 0;
            for (int j = 0; j < N; j++) {
                RowSum += a[i][j];
                ColSum += a[j][i];
            }
        }

        for (int i = 0; i < N; i++) {
            DiaSum1 += a[i][i];
            DiaSum2 += a[i][N - 1 - i];
        }
        if ((RowSum != ColSum) && (DiaSum1 != DiaSum2))
            return false;

        else
            return true;
    }

    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of matrix = ");
        int N = sc.nextInt();
        int a[][] = new int[N][N];
```

```
        System.out.println("Enter the elements for first martix : ");
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                a[i][j] = sc.nextInt();
            }
        }

        if (magic(a, N))
            System.out.println("It is a Magic Matrix");

        else
            System.out.println("Not a magic Matrix");

        sc.close();
    }
}
```

OUTPUT:

```
Enter the size of matrix = 3
Enter the elements for first martix :
2
3
5
6
1
4
5
6
8
It is a Magic Matrix
```

Practical 14:

Write a program showing inheritance example

```
class Box {

   double height, width, depth;

   Box() {
      height = width = depth = -1;
   }

   Box(double h, double w, double d) {
      height = h;
      width = w;
      depth = d;
   }

   double volume() {

      return (width * height * depth);
   }
}

class subBox extends Box {

   double weight;

   subBox() {
      super();
      weight = -1;
   }

   subBox(double w, double h, double d, double m) {
      super(w, h, d); // using super keyword
      weight = m;
   }
}

public class BoxDemo {
```

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

        subBox obj1 = new subBox(1, 2, 3, 4);
        subBox obj2 = new subBox(10, 20, 30, 40);
        subBox obj3 = new subBox();

        System.out.println("The volume is : " + obj1.volume());
        System.out.println("The weight is : " + obj1.weight);

        System.out.println("");

        System.out.println("The volume is : " + obj2.volume());
        System.out.println("The weight is : " + obj2.weight);


        System.out.println("");

        System.out.println("The volume is : " + obj3.volume());
        System.out.println("The weight is : " + obj3.weight);

    }
}
```

OUTPUT:

```
The volume is : 6.0
The weight is : 4.0

The volume is : 6000.0
The weight is : 40.0

The volume is : -1.0
The weight is : -1.0
```

Practical 15:

Write a program showing Final and abstract keyword

```
abstract class A {
   abstract void callme();
   A(){
      System.out.println("hi i am A");
   }
   void callmetoo() {
      System.out.println("This is a concrete method.");
   }
}

class B extends A {
   final int a=2;
   void callme() {
      System.out.println("B's implementation of callme.");
   }



}

class AbstractDemo {
   public static void main(String args[]) {
      A b = new B();
      b.callme();
      b.callmetoo();
   }
}
```

OUTPUT:

```
hi i am A
B's implementation of callme.
This is a concrete method.
```

Practical 16:

Write a programe showing addition and subtraction of complex number

```
public class ComplexNo {

    int r, i;

    ComplexNo(int r, int i) {

        this.r = r;
        this.i = i;
    }

    public void show() {

        System.out.println("The Complex number is :" + r + " + " + i + "i");
    }

    public static ComplexNo add(ComplexNo o1, ComplexNo o2) {

        ComplexNo temp = new ComplexNo(0, 0);

        temp.r = o1.r + o2.r;
        temp.i = o1.i + o2.i;

        return temp;
    }

    public static ComplexNo sub(ComplexNo o1, ComplexNo o2) {

        ComplexNo temp = new ComplexNo(0, 0);

        temp.r = o1.r - o2.r;
        temp.i = o1.i - o2.i;

        return temp;
```

```
    }

    public static void main(String[] args) {

        ComplexNo obj1 = new ComplexNo(1, 2);
        ComplexNo obj2 = new ComplexNo(4, 3);

        System.out.print("The first complex no is : ");
        obj1.show();

        System.out.print("The second complex no is : ");
        obj2.show();

        ComplexNo ans1 = add(obj1, obj2);

        System.out.print("The added complex no is : ");
        ans1.show();

        ComplexNo ans2 = sub(obj1, obj2);

        System.out.print("The Subtracted complex no is : ");
        ans2.show();
    }
}
```

OUTPUT:

```
The first complex no is : The Complex number is :1 + 2i
The second complex no is : The Complex number is :4 + 3i
The added complex no is : The Complex number is :5 + 5i
The Subtracted complex no is : The Complex number is :-3 + -1i
```

Practical 17:

Write a program showing implementation of linked list

```
import java.util.Scanner;

class Node{
    int data;
    Node next;

    Node(int data){
        this.data=data;
        this.next=null;
    }
    void append(Node node){
        Node temp=this;
        while(temp.next!=null){
            temp=temp.next;
        }
        temp.next=node;
    }
    void display(){
        Node temp=this;
        while(temp!=null){
            System.out.println(temp.data);
            temp=temp.next;
        }
    }
    Node delete(int data){
        Node temp = this;
        Node pre=temp;
        if(this.data==data){
            return this.next;
        }
        else{
        while(temp.next.data!=data){
            temp=temp.next;
        }
        temp.next=temp.next.next;
        }
        return this;
```

```
    }
}
class Main{
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        Node node[] = new Node[5];

        for(int i=0;i<5;i++){
            node[i]=new Node(sc.nextInt());
            if(i>=1){
                node[0].append(node[i]);
            }
        }
        System.out.println("display");
        node[0].display();
        int del=sc.nextInt();
        System.out.println("after delete");
        Node n=node[0].delete(del);
        System.out.println("display");

        n.display();
    }
}
```

OUTPUT:

```
3
4
2
6
7
display
3
4
2
6
7
4
after delete
display
3
2
6
7
```

Practical 18:

Write a program showing implementation of Circular linked list

```
import java.util.Scanner;

class Node{
    int data;
    Node next;

    Node(int data){
        this.data=data;
        this.next=null;
    }
    void append(Node node){
        if (this.next == null) {
            this.next = node;
            node.next = this;
        } else {
            Node temp = this;
            while (temp.next != this) {
                temp = temp.next;
            }
            temp.next = node;
            node.next = this;
        }
    }
    void display(){
        Node temp=this;
        System.out.println(temp.data);
        temp=temp.next;
        while(temp!=this){
            System.out.println(temp.data);
            temp=temp.next;
        }


    }
    Node delete(int data){
```

```
        Node temp = this;

        if(temp.data==data){
            while(temp.next!=this){
                temp=temp.next;
            }
            temp.next=this.next;
            return this.next;
        }
        else{
        while(temp.next.data!=data && temp.next!=this){
            temp=temp.next;
        }
        if(temp.next.data==data)
        temp.next=temp.next.next;
        }
        return this;
    }
}
class Circular{
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        Node node[] = new Node[5];

        for(int i=0;i<5;i++){
            node[i]=new Node(sc.nextInt());
            if(i>=1){
                node[0].append(node[i]);
            }
        }
        System.out.println("display");

        node[0].display();
        int del=sc.nextInt();
        System.out.println("display after delete");

        Node n=node[0].delete(del);


        n.display();
```

```
    }
}
```

OUTPUT:

```
3
4
5
6
2
display
3
4
5
6
2
3
display after delete
4
5
6
2
```

Practical 19:

Write a program showing implementation of Doubly linked list

```
import java.util.Scanner;

class Node{
    int data;
    Node next;
    Node prev;

    Node(int data){
        this.data=data;
        this.next=null;
        this.prev=null;
    }
    void append(Node data){
        Node temp=this;
        while(temp.next!=null){
            temp=temp.next;
        }
        temp.next=data;
        data.prev=temp;
    }
    Node delete(int data){
        Node temp = this;

        if(this.data==data){
            this.next.prev=null;
            return this.next;
        }
        else{
```

```
        while(temp.next.data!=data){
            temp=temp.next;
        }
        temp.next=temp.next.next;
        temp.next.next.prev=temp;
        }
        return this;
    }

    void display(){
        Node temp=this;
        while(temp!=null){
            System.out.println(temp.data);
            temp=temp.next;
        }
    }
}
class Doubly{
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        Node node[] = new Node[5];

        for(int i=0;i<5;i++){
            node[i]=new Node(sc.nextInt());
            if(i>=1){
                node[0].append(node[i]);
            }
        }
        System.out.println("display Doubly");
        node[0].display();
        int del=sc.nextInt();
```

```
        Node n=node[0].delete(del);
        System.out.println("display Doubly after deletion");

        n.display();
    }
}
```

OUTPUT:

```
1
2
3
4
5
display Doubly
1
2
3
4
5
1
display Doubly after deletion
2
3
4
5
```

Practical 20:

Write a program showing implementation of Doubly circular linkedList

```
import java.util.Scanner;

class Node{
	int data;
	Node next;
	Node prev;

	Node(int data){
		this.data=data;
		this.next=null;
		this.prev=null;
	}
	void append(Node node){
		Node temp = this;

		if(this.next==null){
			this.next=node;
			node.prev=this;
			node.next=this;
			this.prev=node;
		}
		else{
			while(temp.next!=this){
				temp=temp.next;
			}
			temp.next=node;
			node.prev=temp;
			node.next=this;
			this.prev=node;
		}
	}
	void display(){
		Node temp=this;


		System.out.println(temp.data);
		temp=temp.next;
```

```
        while(temp.next!=this && temp.next!=null){
                System.out.println(temp.data);
                temp=temp.next;
        }
        if(temp.next!=null){
                System.out.println(temp.data);
        }
}
void insertAfter(int data,int pData){
        Node temp = this;

        Node node = new Node(data);

        while(temp.data != pData){
                temp=temp.next;
        }
        node.next=temp.next;
        temp.next=node;
        node.prev=temp;
}
void insertBefore(int data,int aData){
        Node temp = this;

        Node node = new Node(data);

        while(temp.next.data != aData){
                temp=temp.next;
        }
        node.prev=temp;
        node.next=temp.next;
        temp.next=node;
}
Node delete(int data){
        Node temp = this;


        if(this.data==data){
                while(temp.next != this){
                        temp=temp.next;
```

```
                }
                temp.next=this.next;
                this.next.prev=temp;
                return this.next;
            }
            else{
                while(temp.next.data!=data){
                    temp=temp.next;
                }
                temp.next=temp.next.next;
                temp.next.next.prev=temp;
                return this;
            }
        }
}
class DCircular{
        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);

            System.out.println("Enter number of nodes u want add ");
            int n=sc.nextInt();

            Node node[] = new Node[n];
            for(int i=0;i<n;i++){
                node[i]=new Node(sc.nextInt());
                if(i>=1){
                    node[0].append(node[i]);
                }
            }
            System.out.println("display ");
            node[0].display();

            System.out.println("Enter a node u want to delete");
            node[0] = node[0].delete(sc.nextInt());

            System.out.println("Display after deletion of node");
            node[0].display();

            System.out.println("Enter a node after u want to insert");
            int pData = sc.nextInt();
```

```
            node[0].insertAfter(sc.nextInt(),pData);

            System.out.println("Display after insertion of node");
            node[0].display();

            System.out.println("Enter a node before u want to insert");
            int aData = sc.nextInt();

            node[0].insertBefore(sc.nextInt(),aData);

            System.out.println("Display after insertion of node");
            node[0].display();

        }
}
```

OUTPUT:

```
Enter number of nodes u want add
5
2
3
4
5
8
display
2
3
4
5
8
Enter a node u want to delete
3
Display after deletion of node
2
4
5
8
Enter a node after u want to insert
2
3
Display after insertion of node
2
3
4
5
8
Enter a node before u want to insert
8
7
Display after insertion of node
2
3
4
5
7
```

Practical 21:

Write a program to implement access specification using package

- **P1 Package Protection class**

```
package p1;

public class Protection {

   int n = 1;
   private int n_pri = 2;
   protected int n_pro = 3;
   public int n_pub = 4;

   public Protection() {
      System.out.println("Base Constructor (1)");
      System.out.println("n = " + n);
      System.out.println("n_pri = " + n_pri);
      System.out.println("n_pro = " + n_pro);
      System.out.println("n_pub = " + n_pub);
   }
     public static void main(String[] args) {
      Protection p = new Protection();
   }

}
```

- **P1 Package Same Class Non Subcass**

```
package p1;

public class SamePackage {

   SamePackage(){

      Protection p = new Protection();

      System.out.println("Same Package non subclass(1)");
      System.out.println(" n = " + p.n);

      // System.out.println(" n_pri = " + n_pri);
```

```
        System.out.println("n_pro = " + p.n_pro);
        System.out.println("n_pub = " + p.n_pub);
    }

    public static void main(String[] args) {

        SamePackage s = new SamePackage();
    }
}
```

- **P1 Package Same Class Subcass**

```
package p1;

public class Derived extends Protection {

    Derived() {

        System.out.println("Same Package subclass(1)");
        System.out.println("n = " + n);
        // System.out.println("n_pri = " + n_pri);
        System.out.println("n_pro = " + n_pro);
        System.out.println("n_pub = " + n_pub);
    }
     public static void main(String[] args) {
        Derived d = new Derived();
    }
}
```

- **P1 Package Demo class**

```
package p1;

public class Demo {

    public static void main(String[] args) {

        Protection ob1 = new Protection();
        Derived ob2 = new Derived();
        SamePackage ob3 = new SamePackage();
    }
}
```

- **Output**

```
E:\package access code>java p1.Demo
Base Constructor (1)
n = 1
n_pri = 2
n_pro = 3
n_pub = 4
Base Constructor (1)
n = 1
n_pri = 2
n_pro = 3
n_pub = 4
Same Package subclass(1)
n = 1
n_pro = 3
n_pub = 4
Base Constructor (1)
n = 1
n_pri = 2
n_pro = 3
n_pub = 4
Same Package non subclass(1)
 n = 1
n_pro = 3
n_pub = 4
```

- **P2 Package Protection2 class Different Package Subclass**

```
package p2;

class Protection2 extends p1.Protection {

    Protection2() {

        System.out.println("Different package subclass(2)");

        // System.out.println("n = " + n); as other package cannot be access
        // System.out.println("n_pri = " + n_pri); as rivate cannot be access
        System.out.println("n_pro = " + n_pro);
        System.out.println("n_pub = " + n_pub);
    }

    public static void main(String[] args) {
        Protection2 p = new Protection2();
    }
}
```

- **P2 Package Different Package non subclass**

```
package p2;

public class OtherPackagae {

    OtherPackagae() {
        p1.Protection p = new p1.Protection();

        System.out.println("Different Package non subclass(2)");
        System.out.println("n_pub = " + p.n_pub);

        // pri,pro,default canot be access
    }

    public static void main(String[] args) {

        OtherPackagae o = new OtherPackagae();
    }
```

```
}
```

- **P2 Package Demo class**

```
package p2;

public class Demo {

    public static void main(String[] args) {

        Protection2 ob1 = new Protection2();
        OtherPackagae ob2 = new OtherPackagae();
    }
}
```

- **Output**

```
E:\package access code>java p2.Demo
Base Constructor (1)
n = 1
n_pri = 2
n_pro = 3
n_pub = 4
Different package subclass(2)
n_pro = 3
n_pub = 4
Base Constructor (1)
n = 1
n_pri = 2
n_pro = 3
n_pub = 4
Different Package non subclass(2)
n_pub = 4
```

Practical 22:
Write a program to implement user(custom) exception subclass

```
class MyException extends Exception {

    MyException(String msg) {
        super(msg);
    }

}

class Student {
    String Name;
    String Gender;
    String Department;
    int Age;
    double SPI;

    Student(String Name, String Gender, String Department, int Age, double SPI) {

        this.Name = Name;
        this.Gender = Gender;
        this.Department = Department;
        this.Age = Age;
        this.SPI = SPI;
    }

    void checkException() throws MyException {

        if (Department == "IT" && Age == 19) {
            if (SPI > 4.7 && SPI < 5.5) {
                throw new MyException("EXCEPTION CAUGHT");
            }
        }
    }

    public String toString() {
        return "Name = " + Name + "\nGender = " + Gender + "\nDepartment = " + Department +
"\nAge = " + Age
                + "\nSPI = " + SPI;
```

```
    }
}

public class CustExce {

    public static void main(String[] args) {

        Student s1 = new Student("Maitri", "Female", "IT", 19, 4.8);

        try {
            s1.checkException();
        } catch (MyException e) {
            System.out.println(e);
        }

        System.out.println(s1);

        Student s2 = new Student("Jay", "Male", "IT", 19, 4.6);

        try {
            s2.checkException();
        } catch (MyException e) {
            System.out.println(e);
        }

       System.out.println(s2);

    }
}
```

OUTPUT:

```
CAUGHT The Message is INVALID PERSON
Name is : ADI & 19
Name is : RAMDEV & 28
```

Practical 23:

Write a program to implement threads by implementing Runnable class and by extending Thread class

Using Thread class

```
class NewThread extends Thread{
	String name;
	NewThread(String s){
		super("child1");
		start();
	}
	public void run(){
		try{
			for(int i=5;i<10;i++){
				System.out.println(" child1 "+i);
				Thread.sleep(500);
			}
		}catch(InterruptedException e){
			System.out.println("Interrupted child thread");
		}
	}
}
class Thread3{
	public static void main(String[] args) {
		NewThread obj = new NewThread("child1");

		try{
			for(int i=0;i<5;i++){
				System.out.println(i);
				Thread.sleep(1000);
			}
		}catch(InterruptedException e){
			System.out.println("Interrupted main thread");
		}
	}
}
```

OUTPUT:

```
0
 child1 5
 child1 6
1
 child1 7
 child1 8
2
 child1 9
3
4
```

Implement runnable interface

```
class NewThread implements Runnable{
	String name;
	Thread t;

	NewThread(String s){
		name = s;
		t = new Thread(this,"childThread");
		t.start();
	}
	public void run(){
		try{
			for(int i=5;i<10;i++){
				System.out.println(name +" "+i);
				Thread.sleep(500);
			}
		}catch(InterruptedException e){
			System.out.println("Interrupted child thread");
		}
	}
}
class Thread2{
	public static void main(String[] args) {
		NewThread obj = new NewThread("child1");

		try{
			for(int i=0;i<5;i++){
```

```
                        System.out.println(i);
                        Thread.sleep(1000);
                    }
                }catch(InterruptedException e){
                    System.out.println("Interrupted main thread");
                }
        }
}
```

OUTPUT:

```
0
child1 5
child1 6
1
child1 7
child1 8
2
child1 9
3
4
```

Practical 24:

Write a program to implement threads by implementing Producer Consumer problem

```
class Q {
   int n = 0;
   boolean setValue = false;

   synchronized int get() {
      if (!setValue) {
         try {
            wait();
         } catch (InterruptedException e) {
            System.out.println("InterruptedException");
         }
      }
      System.out.println("Consumer : " + n);
      setValue = false;
      notify();
      return n;
   }

   synchronized void put(int n) {
      if (setValue) {
         try {
            wait();
         } catch (InterruptedException e) {
            System.out.println("InterruptedException");
         }
      }
      this.n = n;
      System.out.println("Producer : " + n);
      notify();
      setValue = true;
   }
}

class Producer implements Runnable {
   Q q;
   Thread t;
```

```
    // int n = 0;
    Producer(Q q) {
        this.q = q;
        t = new Thread(this);
        t.start();
    }

    public void run() {
        int i = 0;
        while (true) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;
    Thread t;

    Consumer(Q q) {
        this.q = q;
        t = new Thread(this);
        t.start();
    }

    public void run() {
        while (true) {
            q.get();
        }
    }
}

class PC {
    public static void main(String args[]) {
        Q q = new Q();
        Producer p = new Producer(q);
        Consumer c = new Consumer(q);
    }
}
```

OUTPUT:

```
Consumer : 17
Producer : 18
Consumer : 18
Producer : 19
Consumer : 19
Producer : 20
Consumer : 20
Producer : 21
Consumer : 21
Producer : 22
Consumer : 22
Producer : 23
Consumer : 23
Producer : 24
Consumer : 24
Producer : 25
Consumer : 25
Producer : 26
Consumer : 26
```