**Program 1**

**Check whether the number is fascinating or not. When a number( 3 digits or more ) is multiplied by 2 and 3, and when both these products are concatenated with the original number, then it results in all digits from 1 to 9 present exactly once. There could be any number of zeros and are ignored.**

**Algorithm**

***void main()***

1. Initialize a *object* of class **Fasc** named as '**a**', to call the methods.
2. invoke the **input** method using the 'a' object.
3. Use the '**is\_fasc**' method to check it is a fascinating number and accordingly display the appropriate messages.

***boolean is\_fasc()***

1. The number has to be atleast of 3 digits, so if the number is smaller than 100 return false.
2. Initialize variables n2, n3 storing double and triple of the entered number 'num'.
3. Initialize a integer array '**cnt**' of size of 10 in order to store the count of each digit.
4. Concat num, n2 and n3 as string and parser it back to a integer and store it in '**num**'.
5. Start a **while** loop with condition **num!=0**.
6. Extract the last digit of 'num' and store it in an integer '**dig**'.
7. increment value of ***cnt[dig]*** by 1 as this stores the count of digit.
8. Divide **num** by 10 and store it in **num**.
9. **End While Loop**
10. Start a for loop with value of **x as 1**, condition as **x < 10** and **x=x+1** (*Starting x from as we don't want to check freequency of zero*).
11. if ***cnt[x]*** is not 1 then **return false**, as this means that the digit is not unique and thus violates condition to be fascinating.
12. **End for loop**
13. Return **true** as it is a fascinating number.

***void input()***

1. Initialize the Scanner object with argument System.in to take input.
2. Enter the number into *global variables* '**num**'.

**import** **java.util.\***;

**class** Fasc {

**long** num;

**void** input() {

Scanner sc = **new** Scanner(System.in);

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

num = sc.nextLong();

sc.close();

}

**boolean** is\_fasc() {

**if** (num < 100)

**return** **false**;

**long** n2 = num \* 2, n3 = num \* 3;

**int**[] cnt = **new** **int**[10];

String join = "" + num + n2 + n3;

num = Long.parseLong(join);

**while** (num != 0) {

**int** dig = (**int**) (num % 10);

cnt[dig]++;

num /= 10;

}

**for** (**int** x = 1; x < 10; x++) *// we have to ignore 0s*

**if** (cnt[x] != 1)

**return** **false**;

**return** **true**;

}

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

Fasc a = **new** Fasc();

a.input();

System.out.println("Is " + (a.is\_fasc() ? "" : "not ") + "a Fascinating number");

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| num | long | to store entered number |
| **void main()** |  |  |
| a | Fasc | Object to call the function |
| **boolean isfasc()** |  |  |
| n2 | long | To store double of **num** |
| n3 | long | To store triple of **num** |
| cnt | int[] | Integer array of size 10 to store digit freequency |
| join | String | To store concated num, n2 and n3 |
| dig | int | To store extracted digit |
| **void input()** |  |  |
| sc | Scanner | Object to take User Input |

**Program 2**

**A Kaprekar number is a number in which the sqaure of the number is split into two parts and the sum of those two parts is equal to number. For e.x- 452=2025 -> 20 + 45 == 45, 2972=88209 -> 88 + 209 == 297.**

**Algorithm**

***void main()***

1. Initialize a object of class *Kaprekar* to invoke its methods.
2. Call the function *input()* using the object.

***boolean iskapr(int n)***

1. Initailize a String namesd **nm** which stpores string value of squre of n.
2. Store length of 'nm' in **ln** as a int.
3. Intialize a variable '**first**' and store parsed integer of substring of 'nm' from 0 to 'ln/2'.
4. Intialize a variable '**second**' and store parsed integer of substring of 'nm' from 'ln/2' to last.
5. if the sum of *first* and *second* is equal to n then return true else return false.

***void input()***

1. Intialize the Scanner Object to take input.
2. Input a integer value and store it in integer '**n**'.
3. Pass the integer **n** into *is\_fasc*\* as argument and depending on the return value print if it is a kaprekar number or not.

**import** **java.util.Scanner**;

**class** Kaprekar {

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

Kaprekar a = **new** Kaprekar();

a.input();

}

**void** input() {

Scanner sc = **new** Scanner(System.in);

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

**int** n = sc.nextInt();

**if** (is\_kapr(n))

System.out.println("Is a Kaprekar number");

**else**

System.out.println("Is not a Kaprekar number");

sc.close();

}

**boolean** is\_kapr(**int** n) {

String nm = String.valueOf(n\*n); *// number converted to a string*

**int** ln = nm.length(), first = 0, second = 0;

first = Integer.parseInt(nm.substring(0, ln / 2));

second = Integer.parseInt(nm.substring(ln / 2));

**return** first + second == n;

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **void main()** |  |  |
| a | Kaprekar | Object to call the methods |
| **boolean iskaprekar()** |  |  |
| n | int | **argument** :- variable to test for kaprekar number |
| nm | String | to store string value of sqaure of 'n' |
| ln | int | to store length of string |
| first | int | to store first part of the substring |
| second | int | to store second part of the substring |
| **void input()** |  |  |
| sc | Scanner | To Take User Input |
| n | int | to store entered number |

**Program 3**

**Enter a number and check if it is a smith number. A Smith number is a number in which the sum of digits of a number is equal to sum digits of the prime factors(excluding 1) of that number.**

**Algorithm**

***void main()***

1. Initialize the Object from the class Smith.
2. Call the function *input()*.

***int dig\_sum(int n)***

1. Initialize integer 'r' as 0.
2. Start **while** loop whith condtion **n!=0**.
3. r = r + (n % 10)
4. Divide n by 10 and store it in 'n'.
5. End **while** loop.
6. return r

***boolean check(int num)***

1. Initialize a integer named 'sm' and store sum of digits of num.
2. Inintialize a integer named 'pr\_sm' as 0. This will be used to store sum of digit of prime factors of the number.
3. Initialize int d as 2. This is the first factor to be evaluated.
4. Start **while** loop with condition **num!=1**.
5. Start **if** with condition **num%d==0**.
6. pr\_sm = pr\_sm + (sum of digit of 'd')
7. End **if** and not true then d=d+1
8. End **while**
9. Return *true* if pr\_sm==sm else return *false*.

***void input()***

1. Intialize the Scanner Object to take input.
2. Input a integer value and store it in integer '**n**'.
3. Pass the integer **num** into *check*\* as argument and depending on the return value print if it is a Smith number or not.

**import** **java.util.Scanner**;

**class** Smith {

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

Smith a = **new** Smith();

a.input();

}

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter Number: ");

**int** num = sc.nextInt();

**if** (check(num))

System.out.println("Is a Smith number");

**else**

System.out.println("Is not a Smith number");

sc.close();

}

**boolean** check(**int** num) {

**int** sm = dig\_sum(num); *// sum of digits of the number*

**int** pr\_sm = 0; *// sum of the digits of the prime number*

**int** d = 2;

**while** (num != 1) {

**if** (num % d == 0) {

pr\_sm += dig\_sum(d);

num /= d;

} **else**

d++;

}

**return** pr\_sm == sm;

}

**int** dig\_sum(**int** n) {

**int** r = 0;

**while** (n != 0) {

r += (n % 10);

n /= 10;

}

**return** r;

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **void main()** |  |  |
| a | Smith | Object to call the methods |
| **void input()** |  |  |
| sc | Scanner | Object to take user input |
| num | int | to store entered number |
| **boolean check()** |  |  |
| num | int | **argument** :- number to check if smith |
| sm | int | to store sum of digits of 'num' |
| pr\_sm | int | to store sum of digits of prime factors of 'num' |
| d | int | this is the starting index of prime numbers to be evaluated |
| **int dig\_sum()** |  |  |
| n | int | **argument** :- number whose sum of digits is to be found |
| r | int | to store sum of digits |
| dig | int | to store the extracted digit |

**Program 4**

**A triangular number is formed by the addition of consecutive integers starting with 1. Print all triangular numbers in range of 3 to n.**

**for example:-**

1 + 2 = 3  
1 + 2 + 3 = 6

**Algorithm**

***void main()***

1. Intialize a object of class Traingle.
2. Call the function *input()*.
3. Call the function *print()*.

***void input()***

1. Initialize the Scanner Object to take input.
2. Take input into the variable 'n' as integer.

***boolean check()***

1. Intialize a integer 'sm' as 0 and integer 'x' as 0.
2. Start **for** loop with condition **x<n**.\
3. sm=sm+x
4. if sm>n return *false*
5. else if sm==n return *true*
6. End **for** loop
7. return false, as it is not a triangular number.

***void print()***

1. Start a *for* loop to run from 3 to 'n'
2. Check if the iterated number is triangular then print it.

**import** **java.util.\***;

**public** **class** Triangle {

**int** n;

**boolean** check(**int** n) {

**int** sm = 0;

**for** (**int** x = 0; x < n; x++) {

sm += x;

**if** (sm > n)

**return** **false**;

**else** **if** (sm == n)

**return** **true**;

}

**return** **false**;

}

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Value of n: ");

n = sc.nextInt();

sc.close();

}

**void** print() {

System.out.print("Triangular Numbers:- ");

**for** (**int** x = 3; x <= n; x++)

**if** (check(x))

System.out.print(x + ", ");

}

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

Triangle a = **new** Triangle();

a.input();

a.print();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| n | int | to store entered number for evaluation |
| **void main()** |  |  |
| a | Triangle | Object to call the functions of the class |
| **void input()** |  |  |
| sc | Scanner | to take user input |
| **boolean check()** |  |  |
| sm | int | to store sum of consecutive numbers |
| x | int | a counter variable for the for loop |
| **void input()** |  |  |
| x | int | a counter variable for the for loop |

**Program 5**

**Enter a number and check if it is a Duck number. A number containing one or more zeroes is a duck number.**

**Algorithm**

***void main()***

1. Initialize a object 'a' of class Duck to call its methods.
2. Intialize the Scanner object to take input.
3. Take the number input into integer 'num'
4. Pass the variable 'num' into *isduck()* as argument and based on return value print display if it is a duck number or not.

***boolean isduck(int num)***

1. Start a **for** for with condition **num!=0** and evaluate expression as **num/=10**.
2. if num is a factor of 10,(i.e. contains zero) then return true
3. End **for** loop.
4. return *false* as it not a duck number

**import** **java.util.\***;

**public** **class** Duck {

**boolean** isduck(**int** num) {

**for** (; num != 0; num /= 10)

**if** (num % 10 == 0) *// if 0 is there as a digit then it is a duck number*

**return** **true**;

**return** **false**;

}

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

Duck a = **new** Duck();

Scanner sc = **new** Scanner(System.in);

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

**int** num = sc.nextInt();

System.out.printf("Is%s a Duck Number", (a.isduck(num) ? "" : " not"));

sc.close();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **void main()** |  |  |
| a | Duck | object to call method |
| sc | Scanner | object to take user input |
| num | int | to store entered number |

**Program 6**

**Given a date, check if it is valid or not. It may be assumed that the given date is in range from 01/01/1800 to 31/12/9999.**

**Algorithm**

***void main()***

1. Initialize Object of the class DateVerify to call its method.
2. Take input of the date by calling the *input()* method.
3. Check if the date entered is valid by calling *verify()* and then print the message accordingly.

***void input()***

1. Initialize Scanner object to take input.
2. Store thr entered *date* at index 0 of array **d**.
3. Store thr entered *month* at index 1 of array **d**.
4. Store thr entered *year* at index 2 of array **d**.

***boolean verify()***

1. If the entered year is smaller than the MIN\_DATE[2] or entered year is greater than MAX\_DATE[2] then return **false**.
2. if days are smaller than 1 or month is smaller than 1 or month is greater than 1 then return **false**
3. Initialize a boolean value *isLeap* which is the status if the current is a leap year.
4. If month is February and it a leap year and days are 29 then retuen **true**
5. If days is greater than *DAY\_COUNT[month-1]* then return **false**
6. If *year* == MIN\_DATE[2] and *month* == MIN\_DATE[1] && *days* < MIN\_DATE[0] return **false**
7. If *year* == MAX\_DATE[2] and *month* == MAX\_DATE[1] and *days* > MAX\_DATE[0] return **false**

**import** **java.util.\***;

**public** **class** DateVerify {

**final** **int**[] MIN\_DATE = { 1, 1, 1800 };

**final** **int**[] MAX\_DATE = { 31, 12, 9999 };

**final** **int**[] DAY\_COUNT = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

**public** **int**[] d = **new** **int**[3];

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter day as dd: ");

d[0] = sc.nextInt();

System.out.print("Enter month as mm: ");

d[1] = sc.nextInt();

System.out.print("Enter year as yyyy: ");

d[2] = sc.nextInt();

sc.close();

}

**boolean** verify() {

**if** (!(d[2] >= MIN\_DATE[2] && d[2] <= MAX\_DATE[2]))

**return** **false**;

**if** (d[0] < 1 || d[1] < 1 || d[1] > 12)

**return** **false**;

**boolean** isLeap = d[2] % 4 == 0;

**if** (d[1] == 2 && isLeap && d[0] == 29)

**return** **true**;

**else** **if** (d[0] > DAY\_COUNT[d[1] - 1])

**return** **false**;

*// checking for strict range*

**if** (d[2] == MIN\_DATE[2] && d[1] == MIN\_DATE[1] && d[0] < MIN\_DATE[0])

**return** **false**;

**else** **if** (d[2] == MAX\_DATE[2] && d[1] == MAX\_DATE[1] && d[0] > MAX\_DATE[0])

**return** **false**;

**return** **true**;

}

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

DateVerify a = **new** DateVerify();

a.input();

System.out.printf("The given date %d/%d/%d is %s Date\n", a.d[0], a.d[1], a.d[2],

(a.verify() ? "a Valid" : "an Invalid"));

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| d | int[] | integer array of size 3 to store date |
| MIN\_DATE | int[] | integer array storing minimum possible date |
| MAX\_DATE | int[] | integer array storing maximum possible date |
| DAY\_COUNT | int[] | this stores the day count for 12 months of the year |
| **void main()** |  |  |
| a | DateVerify | object to call method |
| **void input()** |  |  |
| sc | Scanner | to take user Input |

**Program 7**

**Write a program to input a date in ddmmyyyy 8 digit format and print it in 1) dd / mm / yyyy format 2) dd, month name , yyyy format.**

**Algorithm**

***boolean validate()***

1. if day, month or year is smaller than 1 or month is greater than 12 return *false*
2. If month is february and is a leap yaer and days are 29 return *true*
3. if daya > DAY\_COUNT[d[1] - 1] then return *false*
4. return *true*

***void show()***

1. if 'validate()' evaluates false then print "Invalid Date" and **eeturn**
2. Print the date in the required format.

***void input()***

1. Initialize Scanner object to take input.
2. Take date input into the string value 'date'.
3. If length of date is not equal to 8 then print "Invalid Input Format" and **return**.
4. Store d[0] as parsed Integer of date.substring(0, 2)
5. Store d[1] as parsed Integer of date.substring(2, 4)
6. Store d[2] as parsed Integer of date.substring(4, 8)

***void main()***

1. Initialize the object of class DateFormat to invoke methods.
2. Invoke the functions *input* and *show*

**import** **java.util.\***;

**public** **class** DateFormat {

**int**[] d = **new** **int**[3];

**private** **final** **int**[] DAY\_COUNT = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

**private** **final** String[] MONTH\_NAME = {

"January", "February", "March", "April", "May", "June", "July", "August", "September", "October",

"November", "December"

};

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter date as 'ddmmyyyy': ");

String date = sc.next();

**if** (date.length() != 8) {

System.out.println("Invalid Input Format");

System.exit(-1);

}

d[0] = Integer.parseInt(date.substring(0, 2));

d[1] = Integer.parseInt(date.substring(2, 4));

d[2] = Integer.parseInt(date.substring(4, 8));

sc.close();

}

**boolean** validate() {

**if** (d[0] < 1 || d[2] < 1 || d[1] < 1 || d[1] > 12)

**return** **false**;

**if** (d[1] == 2 && d[2] % 4 == 0 && d[0] == 29) *// leap yaer condition*

**return** **true**;

**else** **if** (d[0] > DAY\_COUNT[d[1] - 1])

**return** **false**;

**return** **true**;

}

**void** show() {

**if** (!validate()) {

System.out.println("Invalid Date");

**return**;

}

System.out.printf("%d/%d/%d\n", d[0], d[1], d[2]);

System.out.printf("%d %s, %d\n", d[0], MONTH\_NAME[d[1] - 1], d[2]);

}

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

DateFormat a = **new** DateFormat();

a.input();

a.show();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| d | int[] | Integer array of size 3 to store date |
| DAY\_COUNT | int[] | Stores the number of days of 12 months of the year |
| MONTH\_NAME | String[] | Name of the 12 months of the year |
| **void main()** |  |  |
| a | DateFormat | Object to call its methods |
| **void input()** |  |  |
| sc | Scanner | to take user input |
| date | String | to store enetered date |

**Program 8**

**Write a program to accept the year, month and the weekday name of the 1st day of that month and generate its calendar.**

**Algorithm**

***void main()***

1. Initialize the scanner object to take user input
2. Enter the current year into a integer *y*
3. Enter the month current month name in a variable mName of String typ
4. Enter the starting day of the week and store it in wName of String type
5. make a object of class Calendar and pass values *y, nName, wName* as Constructor parameters.
6. Call the methods *makeCalendar()* and *display()*.

***int indexOf(String[] S, String v)***

1. Start *for* loop with *x=0* and condition *x<S.length* and x=x+1
2. if the value at current index(x) is equal to v then return x
3. End **for** loop.
4. return -1, as no matching element is found.

***void makeCalendar()***

1. Initialize d=0 and start=0
2. Calculate mnIndex as indexOf(MONTH\_NAME, mnName)
3. If it is leap year and month is february then d=19 else it d=DAY\_COUNT[mnIndex].
4. Calculate start as indexOf(DAY\_NAME, startDay)
5. Calculate mnName as MONTH\_NAME[mnIndex]
6. Start **for** loop with *i=0* and condition as *i<d* and i=i+1
7. Calculate rw = (i+start)/7 (This is the row of the matrix for current day)
8. Calculate c; = (i+start)%7 (This is the column of the matrix for current day)
9. Store Cal[rw % 5][cl] = i + 1
10. End **for** loop.

***void display()***

1. Print the relevant markup, i.e. month name, yaer, days of week.
2. Intertate over the Double Dimensional Array *Cal* and print it.(This is the actual Calendar printed)

**import** **java.util.\***;

**class** Calendar {

String mnName, startDay;

**int** y;

**int**[] DAY\_COUNT = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

String[] MONTH\_NAME = {

"January", "February", "March", "April", "May", "June", "July", "August", "September", "October",

"November", "December"

};

String[] DAY\_NAME = {

"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"

};

**int**[][] Cal = **new** **int**[5][7];

Calendar(**int** year, String month, String day) {

y = year;

mnName = month.toLowerCase();

startDay = day.toLowerCase();

}

**private** **int** indexOf(String[] S, String v) {

**for** (**int** x = 0; x < S.length; x++)

**if** (v.equalsIgnoreCase(S[x]))

**return** x;

**return** -1;

}

**void** makeCalendar() {

**int** d = 0, start = 0;

**int** mnIndex = indexOf(MONTH\_NAME, mnName);

**if** (y % 4 == 0 && mnIndex == 1)

d = 29;

**else**

d = DAY\_COUNT[mnIndex];

start = indexOf(DAY\_NAME, startDay);

mnName = MONTH\_NAME[mnIndex];

**for** (**int** i = 0; i < d; i++) {

**int** rw = (i + start) / 7, cl = (i + start) % 7;

Cal[rw % 5][cl] = i + 1;

}

}

**void** display() {

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

System.out.println("\t\t\t " + mnName + " " + y);

System.out.println("\t----------------------------------------------------");

System.out.println("\tSUN\tMON\tTUE\tWED\tTHU\tFRI\tSAT");

System.out.println("\t----------------------------------------------------");

**for** (**int** i = 0; i < Cal.length; i++) {

System.out.print("\t");

**for** (**int** j = 0; j < Cal[0].length; j++) {

**int** vl = Cal[i][j];

System.out.printf("%s\t", vl != 0 ? vl : " ");

}

System.out.println();

}

}

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

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter year : ");

**int** y = sc.nextInt();

System.out.print("Enter month name : ");

String mName = sc.next();

System.out.printf("Enter the week day name of 1st day of %s : ", mName);

String wName = sc.next();

Calendar a = **new** Calendar(y, mName, wName);

a.makeCalendar();

a.display();

sc.close();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| mnName | String | to store the entered month name |
| startDay | String | name of staring weekday |
| y | int | to store entered year |
| DAY\_COUNT | int[] | this stores the number of days in month |
| MONTH\_NAME | Names of the 12 months |  |
| DAY\_NAME | Names of weekdays |  |
| Cal | int[][] | This a 5 by 7 Matrix to store thr calendar |
| **int indexOf()** |  |  |
| S | String[] | This is the string array which is to be searched |
| v | String | Value to search for in the array |
| x | int | this is the index of array where element may be found |
| **void makeCalendar()** |  |  |
| d | int | to store the number of days of the current month |
| start | int | to store the starting day of the week |
| mnIndex | int | this stores the month index |
| mnName | int | this stores the month name |
| **void display()** |  |  |
| i, j | int | counter variables to iterate over the calendar matrix |
| **void main()** |  |  |
| sc | Scanner | object to take user input |
| y | ints | to store entered year |
| mName | String | to store entered month name |
| wName | String | to store starting day of the week |
| a | Calendar | object to call methods |

**Program 9**

**Write a program to accept 2 dates in the string format dd/mm/yyyy and find the difference in days between the 2 dates.**

**import** **java.util.\***;

**public** **class** DateDiff {

**int** d, m, y;

DateDiff(String date) {

**int**[] ind = { date.indexOf('/'), date.lastIndexOf('/') };

d = Integer.parseInt(date.substring(0, ind[0]));

m = Integer.parseInt(date.substring(ind[0] + 1, ind[1]));

y = Integer.parseInt(date.substring(ind[1] + 1));

}

**int** getDaysofMonth() {

**int** res = 0;

**int**[] DAY\_COUNT = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

**if** (y % 4 == 0)

DAY\_COUNT[1]++;

**for** (**int** x = 1; x < m; x++)

res += DAY\_COUNT[x - 1];

**return** res;

}

**int** subtract(DateDiff b) {

**int** d = b.getDaysofMonth() + b.d - (**this**.getDaysofMonth() + **this**.d);

**if** (b.y - **this**.y != 0) {

**int** bg = Math.max(b.y, **this**.y);

**int** sm = Math.min(b.y, **this**.y);

**for** (**int** x = sm; x < bg; x++)

**if** (x % 4 == 0)

d += 366;

**else**

d += 365;

}

**return** d;

}

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

Scanner sc = **new** Scanner(System.in);

System.out.print("Date 1: ");

DateDiff d1 = **new** DateDiff(sc.next());

System.out.print("Date 2: ");

DateDiff d2 = **new** DateDiff(sc.next());

System.out.printf("Difference: %d days", d1.subtract(d2));

sc.close();

}

}

**Program 10**

**Given a time in numbers we can convert it into words. For example :**

**5 : 00 —— five o’clock**

**5 : 10 —— ten minutes past five**

**5 : 15 —— quarter past five**

**5 : 30 —— half past five**

**5 : 40 —— twenty minutes to six**

**5 : 45 —— quarter to six**

**5 : 47 —— thirteen minutes to six**

**Write a program which first inputs two integers, the first between 1 and 12 (both inclusive) and second between 0 and 59 (both inclusive) and then prints out the time they represent, in words.**

**Algorithm**

***void main()***

1. Initialize the Scanner object to take input.
2. Take a String input of time in the format of "hour,minute" into a String varaible *time*
3. Declare ind=time.indexOf(","), to store the index of ',' in the sytring.
4. hr(hours) is parsed integer of substring of time from 0 to *ind*
5. min(minutes) is parsed integer of substring of time from *ind* to end
6. Create a object of class TimeToWord by passing hr, min as constructor parameters and then call the method getName() to print the time in words.

***String getName()***

1. if minutes is 0 then return time in format " o'clock"
2. else if minutes in 15 then return time in format "Quarter past "
3. else if minutes in 30 then return time in format "Half past "
4. else if minutes is 45 then return time in format "Quarter to "
5. else if minutes are smaller than 30 then return time in format " minutes past "
6. else return time in format "<60-minutes> minutes to "

**import** **java.util.\***;

**public** **class** TimeToWord {

**int** h, m;

String[] N = {

"One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Eleven",

"Twelve", "Thirteen", "FourTeen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen", "Twenty",

"Twenty One", "Twenty Two", "Twenty Three", "Twenty Four", "Twenty Five", "Twenty Six",

"Twenty Seven", "Twenty Eight", "Twenty Nine"

};

TimeToWord(**int** hr, **int** min) {

h = hr;

m = min;

}

String getName() {

**if** (m == 0)

**return** String.format("%s o'clock", N[h - 1]);

**else** **if** (m == 15)

**return** String.format("Quarter past %s", N[h - 1]);

**else** **if** (m == 30)

**return** String.format("Half past %s", N[h - 1]);

**else** **if** (m == 45)

**return** String.format("Quarter to %s", N[h]);

**else** **if** (m < 30)

**return** String.format("%s minutes past %s", N[m - 1], N[h - 1]);

**else**

**return** String.format("%s minutes to %s", N[60 - m - 1], N[h]);

}

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

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter Time as hr,min :");

String time = sc.next().trim();

**int** ind = time.indexOf(',');

**int** hr = Integer.parseInt(time.substring(0, ind));

**int** min = Integer.parseInt(time.substring(ind + 1));

System.out.printf("%d:%d %s", hr, min, (**new** TimeToWord(hr, min)).getName());

sc.close();

}

}

## Variable Description

| **Name** | **Type** | **Use** |
| --- | --- | --- |
| **global** |  |  |
| h | int | to store hours |
| m | int | to store minutes |
| N | String[] | Name of numbers from 1 to 29 in a array |
| **void main()** |  |  |
| sc | Scanner | object used to take user input |
| time | String | to store entered time |
| ind | int | stores index of ',' in 'time' |
| hr | int | parsed integer of substring of time from 0 to ind (hours) |
| min | int | parsed integer of substring of time from ind+1 to end (minutes) |

**Program 11**

**Checking saddle point of Matrix MxM. Saddle point is where value is min in rows and max in columns.**

**Algorithm**

***void main()***

1. Initialize the SaddlePoint object.
2. Call the *input()* and *check()* functions.

***void input()***

1. Initialize the scanner object to take the user input
2. Take input of value of **M**, i.e. the dimensions of the sqaure matrix
3. Initialize the 2D Array as M by M Array (reserving memory)
4. Take input into the 2D Matrix and print it as well

***void check()***

1. Initialize a found interger as 0
2. Iterate over the square matrix and check if the value is minimum in row (using *min\_row()* method) and also check it the value in maximum in column (using *max\_col()* method), if it evaluates true then increment value of *found by 1* and print the current index where the saadle point is found, i.e. the row and column.
3. if value of *found* is 0 then print "No Saddle Point found"

***boolean min\_row()***

1. Intialize *min* as A[i][0], here i is the current row
2. Start *for* loop as x=0 and condition x<M and x=x+1
3. min = Math.min(min, A[i][x])
4. End **for** loop
5. if v == min then return true else false (v is the argument value o be checked here)

***boolean max\_col()***

1. Intialize *max* as A[0][i], here i is the current column
2. Start *for* loop as x=0 and condition x<M and x=x+1
3. max = Math.max(max, A[i][x])
4. End *for* loop
5. if v == max then return true else false (v is the argument value o be checked here)

**import** **java.util.Scanner**;

**class** SaddlePoint {

**int** M;

**int**[][] A;

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

SaddlePoint a = **new** SaddlePoint();

a.input();

a.check();

}

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter Order of Square Matrix(M): ");

M = sc.nextInt();

A = **new** **int**[M][M];

System.out.println("Enter Matrix of " + M + " by " + M + ": ");

**for** (**int** x = 0; x < M; x++)

**for** (**int** y = 0; y < M; y++)

A[x][y] = sc.nextInt();

System.out.println("Entered Matrix:-");

**for** (**int** x = 0; x < M; x++) {

**for** (**int** y = 0; y < M; y++)

System.out.print(A[x][y] + "\t");

System.out.println();

}

sc.close();

}

**void** check() {

**int** found = 0;

**for** (**int** x = 0; x < M; x++)

**for** (**int** y = 0; y < M; y++) {

**int** vl = A[x][y];

**if** (min\_row(vl, x) && max\_col(vl, y)) {

found++;

System.out.println("Saddle Point found at " + x + "," + y + " = " + vl);

}

}

**if** (found == 0)

System.out.println("No Saddle Point found");

}

**boolean** min\_row(**int** v, **int** i) {

**int** min = A[i][0];

**for** (**int** x = 0; x < M; x++)

min = Math.min(min, A[i][x]);

**return** min == v;

}

**boolean** max\_col(**int** v, **int** i) {

**int** max = A[0][i];

**for** (**int** x = 0; x < M; x++)

max = Math.max(max, A[x][i]);

**return** max == v;

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| M | int | this is the dimension of the square matrix |
| A | int[][] | this is the main matrix for evaluation |
| **void main(** |  |  |
| a | SaddlePoint | Object to call the functions |
| **void main()** |  |  |
| sc | Scanner | Object to take user input |
| **void check()** |  |  |
| found | int | This is the number of saddle points found |
| x, y | int | This is the counter variable to itertate over the 2D Array |
| **boolean min\_row()** |  |  |
| min | int | this is the minimum value in the row to return |
| **boolea max\_col()** |  |  |
| max | int | this is the maximum value in the column to return |

**Program 12**

**A Wonderous Square is a n by n matrix. \_ It contains Integers from 1 to n2, where each integer appears only once. \_ The sum of integers in any row or column should add upto in any row or column should add upto 0.5\*n(n2+1)**

**Algorithm**

***void main()***

1. Initialize object of WonderSquare to call its function
2. Call the *input()* and *check()* methods.

***void input()***

1. Input the order of the square matrix and store it in 'n'
2. Initialize the A Matrix of order n by n
3. Enter the matrix and print it.
4. Call the *check()* method and if returns it is wonder square and then print it accordingly.

***boolean check()***

1. Initialize a integer array named 'cn' of size 'n2+1'.
2. Start **for** loop with x=0, x<n and x=x+1.
3. Start **for** loop with y=0, y<n and x=x+1
4. Initialize vl=A[x][y]
5. if vl > n<sup2 or vl < 1 then return false
6. cn[vl] = cn[vl] + 1
7. if cn[vl] == 2 then return 2
8. End **for**
9. End **for**
10. Initialize sum = n(n2+1)\*0.5
11. Start **for** loop with x=0, x < n amd x=x+1
12. if row\_sum(x)!=sum or col\_sum(x)!=sum then return false
13. End **for**
14. return **true**

***int col\_sum(int i)***

1. Initialize sum=0
2. Start **for** loop with x=0, x<n and x=x+1
3. sum=sum+A[x][i]
4. End **for** loop
5. return *sum*

***int row\_sum(int i)***

1. Initialize sum=0
2. Start **for** loop with x=0, x<n and x=x+1
3. sum=sum+A[i][x]
4. End **for** loop
5. return *sum*

**import** **java.util.Scanner**;

**class** WonderSquare {

**int** n;

**int**[][] A;

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

WonderSquare a = **new** WonderSquare();

a.input();

a.check();

}

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter Order of Square Matrix(n): ");

n = sc.nextInt();

A = **new** **int**[n][n];

System.out.println("Enter Matrix of " + n + " by " + n + ": ");

**for** (**int** x = 0; x < n; x++)

**for** (**int** y = 0; y < n; y++)

A[x][y] = sc.nextInt();

System.out.println("Entered Matrix:-");

**for** (**int** x = 0; x < n; x++) {

**for** (**int** y = 0; y < n; y++)

System.out.print(A[x][y] + "\t");

System.out.println();

}

sc.close();

**if** (check())

System.out.println("Is a Wonderous Sqaure");

**else**

System.out.println("Is not a Wonderous Sqaure");

}

**boolean** check() {

**int**[] cn = **new** **int**[n \* n + 1]; *// storing count of occurences of numbers from 1 to n^2*

*// ^ Checking if numbers come in range and they occur only once*

**for** (**int** x = 0; x < n; x++)

**for** (**int** y = 0; y < n; y++) {

**int** vl = A[x][y];

**if** (vl > n \* n || vl < 1)

**return** **false**;

**if** (++cn[vl] == 2)

**return** **false**;

}

**int** sum = n \* (n \* n + 1) / 2;

**for** (**int** x = 0; x < n; x++)

**if** (row\_sum(x) != sum)

**return** **false**;

**for** (**int** x = 0; x < n; x++)

**if** (col\_sum(x) != sum)

**return** **false**;

**return** **true**;

}

**int** col\_sum(**int** i) {

**int** sum = 0;

**for** (**int** x = 0; x < n; x++)

sum += A[x][i];

**return** sum;

}

**int** row\_sum(**int** i) {

**int** sum = 0;

**for** (**int** x = 0; x < n; x++)

sum += A[i][x];

**return** sum;

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| n | int | order of the square matrix |
| A | int[][] | the 2D Array |
| **void main()** |  |  |
| a | WonderSquare | object to call its method |
| **void input()** |  |  |
| cn | int[] | a array of size n2+1 to freequency of numbers |
| x, y | int | counter variables to iternate over the 2D Array |
| sum | int | to store the sum of evaluation of row and columns |
| **int col\_sum()** |  |  |
| i | int | **argument** :- index of the current column |
| sum | int | to store column sum |
| **int row\_sum()** |  |  |
| i | int | **argument** :- index of the current row |
| sum | int | to store row sum |

**Program 13**

**Write a program to implement Matric Multiplication.**

**Algorithm**

***void main()***

1. Initialize Matric 'a' using MatrixMul class and take input using *input()* method
2. Initialize Matric 'b' using MatrixMul class and take input using *input()* method
3. Declare C= a.multiply(b), this stores the matrix multiplication resultant
4. display the resultant matrix using *C.print()*

***void input()***

1. Initialize Scanner object to take input
2. Take input in the matrix elements using nested for loops

***void print()***

1. Print the elements of matrix using nested for loops

***MatrixMul multiply(MatrixMul A)***

1. if n is not equal to A.m then print "Matrix Multiplication not Applicable and exit"
2. Initialize the Resultant matrix 'B' of order m by A.n.
3. Start **for** loop with x=0, x<m and x=x+1
4. Start **for** loop with y=0, y<A.n and y=y+1
5. Initialize sum=0
6. Start **for** loop with i=0, i<n and i=i+1
7. sum = sum + arr[x][i] \* A.arr[i][y]
8. End **for** loop
9. Store B.arr[x][y]=sum
10. End **for** loop
11. End **for** loop
12. return B

**import** **java.util.Scanner**;

**class** MatrixMul {

**int**[][] arr;

**int** m, n;

MatrixMul(**int** m1, **int** n1) {

m = m1;

n = n1;

arr = **new** **int**[m][n];

}

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.println("Enter Matrx of " + m + " x " + n + " = ");

**for** (**int** x = 0; x < m; x++)

**for** (**int** y = 0; y < n; y++)

arr[x][y] = sc.nextInt();

}

MatrixMul multiply(MatrixMul A) {

**if** (n != A.m) {

System.out.println("MAtrix Multiplicaton not applicable");

System.exit(-1);

}

MatrixMul B = **new** MatrixMul(m, A.n);

**for** (**int** x = 0; x < m; x++)

**for** (**int** y = 0; y < A.n; y++) {

**int** sum = 0;

**for** (**int** i = 0; i < n; i++)

sum += arr[x][i] \* A.arr[i][y];

B.arr[x][y] = sum;

}

**return** B;

}

**void** print() {

**for** (**int** x = 0; x < arr.length; x++) {

**for** (**int** y = 0; y < arr[0].length; y++)

System.out.print(arr[x][y] + "\t");

System.out.println();

}

}

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

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter m,n of the Matrix 1: ");

MatrixMul a = **new** MatrixMul(sc.nextInt(), sc.nextInt());

a.input();

System.out.print("Enter m,n of the Matrix 2: ");

MatrixMul b = **new** MatrixMul(sc.nextInt(), sc.nextInt());

b.input();

MatrixMul C = a.multiply(b);

System.out.println("Multiplied Matrix: ");

C.print();

sc.close();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| m | int | number of rows |
| n | int | number of columns |
| arr | int[][] | array of dimension m by n |
| **void input()** |  |  |
| sc | Scanner | object to take user input |
| **MatrixMul multiply()** |  |  |
| A | MatrixMul | **argument** :- the second matrix |
| B | MatrixMul | the resultant of matrix multiplication |
| **void main()** |  |  |
| sc | Scanner | to take user input |
| a | MatrixMul | to store first matrix |
| b | MatrixMul | to store second matrix |
| C | MatrixMul | to store resultant matrix of multiplication |

**Program 14**

**A square matrix is the matrix in which number of rows equals the number of columns. Thus, a matrix of order n\*n is called a Square Matrix. Write a program in Java to create a double dimensional array of size nxn matrix form and fill the numbers in a circular fashion (anticlock-wise) with natural numbers from 1 to n2, taking n as an input. The filling of the elements should start from outer to the central cell.**

For example: - **n=5**

1 16 15 14 13

2 17 24 23 12

3 18 25 22 11

4 19 20 21 10

5 6 7 8 9

**Algorithm**

***void main()***

1. Initialize the object of class *CircleMat*
2. Call the methods *input()*, *fillMat()* and *display()* successively.

***void input()***

1. Initialize Scanner object to take input
2. Enter value of n into the variable 'n'.

***void fillMat()***

1. Initialize the 2D Array *'A'* as a n by n matrix
2. Initialize value of a=0 and x=1
3. Start **while** loop with condition *x <= n2*
4. Store b = n-1-a
5. Add a **for** loop with *i=a*, *i<=b* and i=i+1 and do A[i][a] = x++ (this prints from left top to left bottom)
6. Add a **for** loop with *i=a+1*, *i<=b* and i=i+1 and do A[b][i] = x++ (this prints pattern from left bottom to right bottom)
7. Add a **for** loop with *i=b-1*, *i>=a* and i=i-1 and do A[i][b] = x++ (this prints pattern from right bottom to right top)
8. Add a **for** loop with *i=b-1*, *i>=a+1* and i=i-1 and do A[a][i] = x++ (this prints pattern from right top to back to left top)

***void display()***

1. Iterate over the 2D Array and print it and this will the *required circular pattern*.

**import** **java.util.\***;

**public** **class** CircleMat {

**int** n;

**int**[][] A;

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Value of n: ");

n = sc.nextInt();

sc.close();

}

**void** fillMat() {

A = **new** **int**[n][n];

**int** a = 0, x = 1;

**while** (x <= n \* n) {

**int** b = n - 1 - a;

**for** (**int** i = a; i <= b; i++)

A[i][a] = x++;

**for** (**int** i = a + 1; i <= b; i++)

A[b][i] = x++;

**for** (**int** i = b - 1; i >= a; i--)

A[i][b] = x++;

**for** (**int** i = b - 1; i >= a + 1; i--)

A[a][i] = x++;

a++;

}

}

**void** display() {

System.out.println("Circular Matrix :- Anti Clockwise");

**for** (**int** x = 0; x < n; x++) {

**for** (**int** y = 0; y < n; y++)

System.out.print(A[x][y] + "\t");

System.out.println();

}

}

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

CircleMat a = **new** CircleMat();

a.input();

a.fillMat();

a.display();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| n | int | to store the dimensions of the square matrix |
| A | int[][] | the matrix in which the pattern to be printed will be stored |
| **void main()** |  |  |
| a | CircleMat | object to call it's method |
| **void display()** |  |  |
| x, y | int | counter variables to iterate over the matrix |
| **void input()** |  |  |
| sc | Scanner | object to take user input |
| **void fillMat()** |  |  |
| x | int | this is the value which will be stored in the matrix |
| a, b | int | indices to help us give printing location |

**Program 15**

**Given a square matrix M[][] of order 'n'. The maximum value possible for 'n' is 10. Accept three different characters from the keyboard and fill the array according to the output shown in the examples. If the value of n exceeds 10 then an appropriate message should be displayed.**

Input: First Character '\_'

Second Character '?'

Third Character '#'

Output: # \_ \_ #

? # # ?

? # # ?

# \_ \* #

**Algorithm**

***void main()***

1. Initialize object of *MatPattern* class to call its function.
2. Call the functions *input()* and *pattern()*

***void input()***

1. Enter of value of 'n' and store it.
2. If value of n is not in range of *1 to 10* then print "SIze out of Range" and exit
3. Enter the three characters from which the pattern is to be printed

***void pattern()***

1. Store *size=n*
2. Initialize an array of size 3 named 'con' to store constion to print characters
3. Print a new line
4. Start **for** loop as y=0, y<size and y=y+1
5. Start **for** loop as x=0, x<size and x=x+1
6. Initialize *r\_x*=n-x, this is the index from the right side
7. con[0] = (x > y and r\_x > y) or (x < y and r\_x < y)
8. con[1] = (x < y and r\_x > y) or (x > y and r\_x < y)
9. con[2] = y == x or y == r\_x
10. Start **for** loop as i=0, i<3 and i=i+1
11. if con[i] is true then print ch[i]+" "
12. End **for** loop
13. End **for** loop
14. Print a new line
15. End **for** loop

**import** **java.util.\***;

**public** **class** MatPattern {

**char**[] ch = **new** **char**[3];

**int** n;

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter Size :");

n = sc.nextInt();

**if** (n < 1 || n > 10) {

System.out.println("Size out of range");

System.exit(-1);

}

**for** (**int** x = 0; x < 3; x++) {

System.out.print("Character " + (x + 1) + " :");

ch[x] = sc.next().charAt(0);

}

sc.close();

}

**void** pattern() {

**int** size = n;

n--;

**boolean**[] con = **new** **boolean**[3]; *// condition to print*

System.out.println();

**for** (**int** y = 0; y < size; y++) {

**for** (**int** x = 0; x < size; x++) {

**int** r\_x = n - x; *// reverse index of x*

con[0] = (x > y && r\_x > y) || (x < y && r\_x < y);

con[1] = (x < y && r\_x > y) || (x > y && r\_x < y);

con[2] = y == x || y == r\_x;

**for** (**int** i = 0; i < 3; i++)

**if** (con[i])

System.out.print(ch[i] + " ");

}

System.out.println();

}

}

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

MatPattern a = **new** MatPattern();

a.input();

a.pattern();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| ch | char[] | the 3 characters on which the pattern will be based |
| n | int | the dimension of the 2D Array(sqaure) to be printed |
| **void input()** |  |  |
| sc | Scanner | object to take user input |
| **void pattern()** |  |  |
| size | int | to store the size of the sqaure matrix |
| con | int[] | condition if the 3 characters are to be printed |
| x, y | int | counter variables to iterate over the matrix |
| i | int | counter variable to print the 3 charcaters |
| **void main()** |  |  |
| a | MatPattern | object to call its methods |

**Program 16**

**A class Rearrange has been defined to modify a word by bringing all the vowels in the word at the beginning followed by the consonants.**

**Algorithm**

***void readword()***

1. Initialize a scanner object to take input.
2. Enter a string(word) and convert it into upper case and store it in global variable 'wrd'

***void freq\_vow\_count()***

1. Start **for** loop with *x=0, x<wrd.length(), x=x+1*.
2. If the charcter at the curreny index 'x' is a vowel then incremrnt *vow\_cn* by 1 else increment *oth\_cn* by 1.
3. End **for** loop
4. Print the values of 'vow\_cn' and 'oth\_cn'.

***void arrange()***

1. Initialize a StringBuffer named 'vow' to store the vowels
2. Initialize a StringBuffer named 'oth' to store other characters
3. Start **for** loop with x=0, x<wrd.length() and x=x+1
4. If the character at current position is a vowel then append the charcter to 'vow' else append it to 'oth'
5. End **for** loop
6. newwrd = vow.toString() + con.toString()

***void display()***

1. Print the Original word
2. Print the Rearranged word

***boolean is\_vowel(char val)***

1. Initialize *vw* as "AEIOU"
2. return vw.indexOf(val)!=-1

***void main()***

1. Initialize the object of class VowelShift to call its functions
2. Call the function readword() to input the word
3. Call the function freq\_vow\_con() to count the freequency of vowels and to display them
4. Call the function arrange() and then display() to rearrange and then display the results.

**import** **java.util.\***;

**class** VowelShift {

String wrd, newwrd;

**int** vow\_cn = 0, oth\_cn = 0;

VowelShift() {

}

**void** readword() {

Scanner sc = **new** Scanner(System.in);

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

wrd = sc.next().toUpperCase();

sc.close();

}

**void** freq\_vow\_con() {

**for** (**int** x = 0; x < wrd.length(); x++)

**if** (is\_vowel(wrd.charAt(x)))

vow\_cn++;

**else**

oth\_cn++;

System.out.println("Vowel count: " + vow\_cn);

System.out.println("Consonant count: " + oth\_cn);

}

**void** arrange() {

StringBuffer vw = **new** StringBuffer(""); *// vowels*

StringBuffer oth = **new** StringBuffer(""); *// others*

**for** (**int** x = 0; x < wrd.length(); x++) {

**char** c = wrd.charAt(x);

**if** (is\_vowel(c))

vw.append(c);

**else**

oth.append(c);

}

newwrd = vw.toString() + oth.toString();

}

**void** display() {

System.out.println("Original Word: " + wrd);

System.out.println("Rearranged Word: " + newwrd);

}

**boolean** is\_vowel(**char** val) {

String vw = "AEIOU";

**return** vw.indexOf(val) != -1;

}

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

VowelShift a = **new** VowelShift();

a.readword();

a.freq\_vow\_con();

a.arrange();

a.display();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| wrd | String | to store the entered word |
| newwrd | String | to store changed(final) word |
| vow\_cn | int | to store vowel freequency |
| oth\_cn | int | to store other characters freequency |
| **void readword()** |  |  |
| sc | Scanner | object to take user input |
| **void freq\_vow\_con()** |  |  |
| x | int | counter variable to iterate over the string value |
| **void arrange()** |  |  |
| vw | StringBuffer | to store extracted vowels |
| oth | StringBuffer | to store other extracted characters |
| **void main()** |  |  |
| a | VowelShift | to call the objects its class |

**Program 17**

**Count the frequency of Words starting with Capital Letter in a sentence.**

**Algorithm**

***void input()***

1. Initialize the scaner object to take input from the user
2. Enter the sentence and store it in 'sent'

***void main()***

1. Initialize the object of class Capital to call its methods
2. Call the input and display functions

***void display()***

1. Initialize a string tokenizer object with ' ' and '.' as delimiters.
2. Start a **while** loop with condtition *st.hasMoreTokens()*
3. if the first charcter of the current token(word) is in upper case then then increment *freq* by 1
4. End **while** loop
5. Print the sentence entered
6. Print the frequency of Words starting with capital letters

**import** **java.util.\***;

**public** **class** Capital {

String sent;

**int** freq;

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter Sentence: ");

sent = sc.nextLine();

sc.close();

}

**void** display() {

StringTokenizer st = **new** StringTokenizer(sent, " .");

**while** (st.hasMoreTokens())

**if** (Character.isUpperCase(st.nextToken().charAt(0)))

freq++;

System.out.println("Sentence: " + sent);

System.out.println("Freequency: " + freq);

}

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

Capital a = **new** Capital();

a.input();

a.display();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| sent | String | to store entered sentence |
| freq | int | to store freequeny of words starting with capital letters |
| **void input()** |  |  |
| sc | Scanner | object to take user input |
| **void display()** |  |  |
| st | StringTokenizer | object used to extract words from the senetence |
| **void main()** |  |  |
| a | Capital | Object to call its classes |

**Program 18**

**A class SwapSort to swap first and last characters of a word and print it, and also sort the alphabets and print it.**

**Algorithm**

***void readstring()***

1. Initialize the scanner object to take user input
2. Store the entered word in 'wrd'
3. Store the length of 'wrd' in 'len'

***void swapchar()***

1. Make a StringBuffer 'a' from 'wrd'
2. a.setCharAt(0, wrd.charAt(len - 1))
3. a.setCharAt(len - 1, wrd.charAt(0))
4. swapwrd=a.toString()

***void swapword()***

1. Initialize the character array 'arr' from String 'wrd'
2. Start **for** loop with x=0, x<len-1 and x=x+1
3. Start **for** loop with y=0, y<len-1-x and y=y+1
4. Start **if** with condition *arr[y]>arr[y+1]*.
5. Store a character 'tmp' as arr[y]
6. arr[y] = arr[y + 1]
7. arr[y + 1] = tmp
8. End **if**
9. End **for** loop
10. End **for** loop

***void display()***

1. Print the original entered word
2. Print the swapped word and sorted word

***void main()***

1. Initailize the object of class *SwapSort*
2. Call the method readstring(), swapchar(), sortword() and display()

**import** **java.util.\***;

**class** SwapSort {

String wrd, swapwrd, sortwrd;

**int** len;

**void** readstring() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter a word: ");

wrd = sc.next();

len = wrd.length();

sc.close();

}

**void** swapchar() {

StringBuffer a = **new** StringBuffer(wrd);

a.setCharAt(0, wrd.charAt(len - 1));

a.setCharAt(len - 1, wrd.charAt(0));

swapwrd = a.toString();

}

**void** sortword() {

**char**[] arr = wrd.toCharArray();

**for** (**int** x = 0; x < len - 1; x++)

**for** (**int** y = 0; y < len - 1 - x; y++)

**if** (arr[y] > arr[y + 1]) {

**char** tmp = arr[y];

arr[y] = arr[y + 1];

arr[y + 1] = tmp;

}

sortwrd = String.valueOf(arr);

}

**void** display() {

System.out.println("Word: " + wrd);

System.out.println("Swapped Word: " + swapwrd);

System.out.println("Sorted Word: " + sortwrd);

}

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

SwapSort a = **new** SwapSort();

a.readstring();

a.swapchar();

a.sortword();

a.display();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| wrd | String | to store entered word |
| swapwrd | String | to store swapped word |
| sortwrt | String | to store swapped word |
| len | int | to store the length of the word |
| **void readstring()** |  |  |
| sc | Scanner | object to take user input |
| **void swapchar()** |  |  |
| a | StringBuffer | temporary string object to interchange characters |
| **void sortword()** |  |  |
| arr | char[] | to store the string as character arry to sort |
| x, y | int | the counter variable to sort the array |
| tmp | char | temporary value to interchange |
| **void main()** |  |  |
| a | SwapSort | Object to call its methods |

**Program 19**

**Print String Fibonacci Series, starting with “a” and “b”.**

Example :- a, b, ba, bab, babba, babbabab

**Algorithm**

***void main()***

1. Initialize a object *StrFib*.
2. Call the functions accept() and series()

***void accept()***

1. Initialize the scanner object to take user input
2. Enter value of '**n**'

***void series()***

1. Print 'x', 'y'
2. Start **for** loop with i=2, i<=n and i=i+1
3. z=y+x
4. Print z
5. x=y
6. y=z
7. End **for** loop

**import** **java.util.\***;

**public** **class** StrFib {

String x = "a", y = "b", z;

**int** n;

**void** accept() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter value of 'n': ");

n = sc.nextInt();

sc.close();

}

**void** series() {

System.out.println("Series: ");

System.out.print(x + ", " + y + ", ");

**for** (**int** i = 2; i <= n; i++) {

z = y + x;

System.out.print(z + ", ");

x = y;

y = z;

}

}

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

StrFib a = **new** StrFib();

a.accept();

a.series();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| x | String | first entry point of fibonacci series as "a" |
| y | String | second entry point of fibonacci series as "b" |
| z | String | to store the next item in the series |
| n | int | index till whicht the series shall be printed |
| **void accept()** |  |  |
| sc | Scanner | object to take user input |
| **void main()** |  |  |
| a | StrFib | object to call the functions of the class |
| **void series()** |  |  |
| i | int | counter variable to generate the series |

**Program 20**

**Exchange the first and last letters of words in a sentence.**

**Example: -**

Input: Java is the one of the most hated language

Output: aavJ si eht eno fo ehh tosm dateh eanguagl

**Algorithm**

***void main()***

1. create a object of class Exchange
2. Call the functions input(), exfirstlast() and display

***void input()***

1. Initialize Scanner object to take user input
2. Enter the sentence in 'sent'
3. Store length of 'sent' in 'size'

***String exchange()***

1. Initialize a StringBuffer 'a'
2. a.setCharAt(0, s.charAt(s.length() - 1))
3. a.setCharAt(s.length() - 1, s.charAt(0))
4. return a.toString()

***void exfirstlast()***

1. Initialize a StringTokenizer object to get the words as 'st'.
2. Initialize 'str' as a StringBuffer object to store the output
3. Start **while** loop with condition st.hasMoreTokens().
4. Store s=st.nextToken()
5. str.append(exchange(s)+" ")
6. End **for** loop
7. rev=str.toString()

***void display()***

1. Print the original sentence
2. Print the changed sentence

**import** **java.util.\***;

**public** **class** Exchange {

String sent, rev;

**int** size;

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter Sentence: ");

sent = sc.nextLine();

size = sent.length();

sc.close();

}

**void** exfirstlast() {

StringTokenizer st = **new** StringTokenizer(sent, " .");

StringBuffer str = **new** StringBuffer();

**while** (st.hasMoreTokens()) {

String s = st.nextToken();

str.append(exchange(s) + " ");

}

rev = str.toString();

}

**void** display() {

System.out.println("Sentence: " + sent);

System.out.println("Changed Sentence: " + rev);

}

**private** String exchange(String s) {

StringBuffer a = **new** StringBuffer(s);

a.setCharAt(0, s.charAt(s.length() - 1));

a.setCharAt(s.length() - 1, s.charAt(0));

**return** a.toString();

}

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

Exchange a = **new** Exchange();

a.input();

a.exfirstlast();

a.display();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **Global** |  |  |
| Sent | String | to store the entered string |
| Rev | String | to store the reversed(word order) string required |
| Size | int | the size of the word entered |
| **void input()** |  |  |
| Sc | Scanner | object to take user input |
| **void exfirstlast()** |  |  |
| St | StringTokenizer | object to extract tokens(words) from the sentence |
| Str | StringBuffer | to store the new resultant string |
| **String exchange()** |  |  |
| A | StringBuffer | to make and store the resultant string |
| **void main()** |  |  |
| a | Exchange | object to call its functions |

**Program 21**

**Shift a Matrix up by one row.**

## Algorithm

#### void main()

1. Initialize the Scanner object to take input
2. Enter the dimensions of the matrix and pass it as the constructor parameters in the object initialization of class Shift
3. Call the methods as input(), cyclic(a), display()

#### void input()

1. Initialize the scanner object to take input
2. Take input of the Matrix 'mat' which is m by n using nested for loops.

#### void cyclic(Shift A)

1. Initailize a matrix 'mt' of the same dimensions as that of 'mat'
2. Start **for** loop as x=0, x<m and x=x+1
3. Start **for** loop as y=0, y<n and y=y+1
4. mt[(x-1)%(m+1)]=A.mat[x][y]
5. End **for** loop
6. End **for** loop

#### void display()

1. Print the shifted array using nested for loops for iterating over the matrix to print individual cells

**import** **java.util.\***;

**class** Shift {

**int**[][] mat;

**int** m, n;

Shift(**int** mm, **int** nn) {

m = mm;

n = nn;

mat = **new** **int**[m][n];

}

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.println("Enter Matrix of " + m + " x " + n + " :");

**for** (**int** x = 0; x < m; x++)

**for** (**int** y = 0; y < n; y++)

mat[x][y] = sc.nextInt();

}

**void** cyclic(Shift A) {

**int**[][] mt = **new** **int**[m][n];

**for** (**int** x = 0; x < m; x++)

**for** (**int** y = 0; y < n; y++)

mt[(x - 1) % (m + 1)][y] = A.mat[x][y];

mat = mt;

}

**void** display() {

System.out.println("Shifted Arary :-");

**for** (**int** x = 0; x < m; x++) {

**for** (**int** y = 0; y < n; y++)

System.out.print(mat[x][y] + "\t");

System.out.println();

}

}

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

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter m,n for Matrix: ");

Shift a = **new** Shift(sc.nextInt(), sc.nextInt());

a.input();

a.cyclic(a);

a.display();

sc.close();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| mat | int[][] | the matrix to be shifted by one row |
| m, n | int | dimensions of the matrix |
| **void input()** |  |  |
| sc | Scanner | object to take user input |
| x, y | int | counter variables used to enter value into the matrix |
| **void cyclic()** |  |  |
| mt | int[][] | the matrix to store the shifted matrix |
| x, y | int | counter variables to iterate over the matrix |
| **void display()** |  |  |
| x, y | int | counter variables to iterate over the matrix to print it |
| **void main()** |  |  |
| sc | Scanner | object to take user input |
| a | Shift | object to call the methods of the class |

**Program 22**

**A bank intends to design a program to display the denomination of an input amount, up to 5 digits. The available denomination with the bank are of rupees 2000, 500, 200, 100, 50, 20, 10 and 1.**

**Algorithm**

***void main()***

1. Create the object of the current class.
2. From the objects call the functions input(), name() and denomination()

***void input()***

1. Initialize the scanner object to take input
2. Enter amount in long integer 'amt'
3. if the length of amt is greater than 5 or less than 0 then print "Invalid Input" and then exit the program

***void name()***

1. Initialize a StringBuffer to store the name of amount in words.
2. Initialize a String array containing name of digits from 0 to 9
3. Store tmp=amt, this is to temporarily store and process 'amt' variable without loosing its value
4. Start **while** loop with condition *tmp!=0*
5. Store dig = tmp%10, to extract the last digit;
6. nm.insert(0, names[(int) dig] + " ")
7. tmp = tmp / 10
8. End **while** loop
9. print the value of nm, this is the name of amount in words

***void denomination()***

1. Initialize a *Long* array 'order' containing the denominations which you want your amount to split into in descending order.
2. Start **for** loop with x=0, x<order.length and x=x+1
3. Store v=amt/order[x]
4. Start **if** with condition *v!=0*
5. Store pr=order[x]\*v
6. Print order[x] + " \* " + v + " = " + pr, (this is denomination breakup)
7. amt=amt-pr
8. End **if**
9. End **for**

**import** **java.util.\***;

**public** **class** Bank {

**long** amt;

**void** input() {

Scanner sc = **new** Scanner(System.in);

System.out.print("Enter Amount: ");

amt = sc.nextLong();

**if** (String.valueOf(amt).length() > 5 || amt < 0) {

System.out.println("Invalid Input");

System.exit(-1);

}

sc.close();

}

**void** name() {

StringBuffer nm = **new** StringBuffer();

String[] names = { "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine" };

**long** tmp = amt;

**while** (tmp != 0) {

**long** dig = tmp % 10;

nm.insert(0, names[(**int**) dig] + " ");

tmp /= 10;

}

System.out.println(nm.toString());

}

**void** denomination() {

**long**[] order = { 2000, 500, 200, 100, 20, 10, 5, 2, 1 };

**for** (**int** x = 0; x < order.length; x++) {

**long** v = amt / order[x];

**if** (v != 0) {

**long** pr = order[x] \* v;

System.out.println(order[x] + " \* " + v + " = " + pr);

amt -= pr;

}

}

}

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

Bank a = **new** Bank();

a.input();

a.name();

a.denomination();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **global** |  |  |
| amt | long | to store netered amount |
| **void input()** |  |  |
| sc | Scanner | object to take user input |
| **void name()** |  |  |
| nm | StringBuffer | to store the name of amount |
| names | String[] | string array storing names of digits from 0 to 9 |
| tmp | int | to temporarily store the amount value |
| **void denomination()** |  |  |
| order | long[] | stores the order of decreasing denominatory notes |
| v | long | number of notes of a denomination |
| **void main()** |  |  |
| a | Bank | object to call its method |

**Program 23**

**Make a Linked list to Enter numbers and sort them.**

**Algorithm**

**Class Node**

***public Node(int d, Node next)***

1. data=d, data
2. link=next, to store the link to the next node

***public void setLink(Node n)***

1. link=n, set the link to the next node

***public void setData(int d)***

1. data=d, set the value of current Node

***public int getData()***

1. return data(value of the current node)

***public Node getLink()***

1. return link

**Class linkedList**

***boolean is\_empty()***

1. return true if start is equal to null else return false

***void insert(int val)***

1. Initialize a Node nptr with value 'val' and link=null
2. *If* isEmpty() is true then *start=nptr*\*, i.e. if the starting element of the list is null then new pointer created wiil have to be our starting pointer.
3. *Else If* val<=start.getLik() then *nptr.setLink(start)* and *start=nptr*. So if the value is smaller than the value in the start node then nptr should be our first node.
4. Start **else**
5. Store Node ptr=start.getLink()
6. Store Node save=start (the previous node)
7. Start **while** loop with condition ptr!=null
8. Start **if** with val >= save.getDeta() and val <= ptr.getDeta() (condition for the required location)
9. save.setLink(nptr) (connect the previous node to the new node)
10. nptr.setLink(ptr) (connect the new node to the current node)
11. return
12. End **if**
13. Start **else**
14. save=ptr (storing previous node for later use)
15. ptr=ptr.getLink()
16. End **else**
17. End **while** loop
18. save.setLink(nptr) (connecting the previous node to the new node)
19. End **else**

***publlic void show()***

1. Initialize a Node ptr=start
2. Start a **for** loop with '', ptr.getLink()!=null and ptr=ptr.getLink()
3. Print ptr.getata()
4. End **for** loop
5. Print ptr.getata()

**Class LinkSort**

***void main()***

1. Initialize the Scanner object to take user input.
2. Create a linkedList named 'list'
3. Start **for** loop with x=1, x<=5 and x=x+1
4. Enter a number into 'num' which will be inserted in the linkedList 'list'
5. list.insert(num)
6. Prin the inserted number
7. End **for** loop
8. list.show()

**import** **java.util.\***;

**class** Node {

**protected** **int** data = 0;

**protected** Node link = **null**;

**public** Node() {

}

**public** Node(**int** d, Node next) {

data = d;

link = next;

}

**public** **void** setLink(Node n) {

link = n;

}

**public** **void** setData(**int** d) {

data = d;

}

**public** **int** getDeta() {

**return** data;

}

**public** Node getLink() {

**return** link;

}

}

**class** linkedList {

**protected** Node start = **null**;

**public** **boolean** isEmpty() {

**return** start == **null**;

}

**public** **void** insert(**int** val) {

Node nptr = **new** Node(val, **null**);

**if** (isEmpty()) {

start = nptr;

} **else** **if** (val <= start.getDeta()) {

nptr.setLink(start);

start = nptr;

} **else** {

Node ptr = start.getLink(), save;

save = start;

**while** (ptr != **null**) {

**if** (val >= save.getDeta() && val <= ptr.getDeta()) {

save.setLink(nptr);

nptr.setLink((ptr));

**return**;

} **else** {

save = ptr;

ptr = ptr.getLink();

}

}

save.setLink(nptr);

}

}

**public** **void** show() {

System.out.println("Sorted LinkedList");

Node ptr = start;

**for** (; ptr.getLink() != **null**; ptr = ptr.getLink())

System.out.printf("%d --> ", ptr.getDeta());

System.out.printf("%d !!!!\n", ptr.getDeta());

}

}

**public** **class** LinkSort {

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

Scanner sc = **new** Scanner(System.in);

linkedList list = **new** linkedList();

System.out.println("Enter Numbers to insert in list: ");

**for** (**int** x = 1; x <= 5; x++) {

**int** num = sc.nextInt();

list.insert(num);

System.out.printf("Inserted: %s\n", num);

}

list.show();

sc.close();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **Class Node** |  |  |
| **global** |  |  |
| data | int | value to be stored in the current node |
| link | Node | pointer for the next node |
| **Class linkedList** |  |  |
| **global** |  |  |
| start | Node | stores the pointer to starting node of the linked list |
| **public void insert()** |  |  |
| val | int | **argument** :- the value to be inserted in linked list |
| nptr | Node | node made using the value 'val' to insert in the list |
| ptr | Node | used as a temporary pointer to traverse the linked list |
| save | Node | to store pointer of the previous Node |
| **public void show()** |  |  |
| ptr | Node | used as a temporary pointer to traverse the linked list |
| **Class LinkSort** |  |  |
| **void main()** |  |  |
| sc | Scanner | object to take user input |
| list | linkedList | object to store our linked auto-sorted linked list |
| num | int | to store enetred number |

**Program 24**

**Parenthesis matching using Stack.**

**Algorithm**

**Class Node**

***public Node(int d, Node next)***

1. data=d, data
2. link=next, to store the link to the next node

***public void setLink(Node n)***

1. link=n, set the link to the next node

***public void setData(int d)***

1. data=d, set the value of current Node

***public int getData()***

1. return data(value of the current node)

***public Node getLink()***

1. return link

**Class Stack**

***public void push(int val)***

1. Create a Node nptr with value of 'val' and 'top' as its link(next elemenet)
2. top=nptr

***public int pop()***

1. If isEmpty() returns true then throw a *Exception* with message "Stack Underflow"
2. Start **else**
3. Store val=top.getData()
4. top=top.getLink()
5. return val
6. End **else**

***public boolean isEmpty()***

1. if top==null then return true else return false

***public void show()***

1. Initailize a Node ptr=top
2. Start **while** loop with condition *ptr!=null*
3. Print ptr.getData()
4. ptr=ptr.getLink() (going to the next element in the stack)
5. End **while** loop

**Class MatchParenStack**

***void main()***

1. Initaialize a Scanner object to accept user input
2. Enter the *expression* for parenthesis matching into String 'val'
3. Initialize a Stack object '*stk*'
4. Start **for** loop with x=0, x<val.length() and x=x+1
5. Store ch=val.charAt(x)
6. if ch=='(' then stk.push(x)
7. else if ch==')' then print the curent index where it has been found and position from which it has been matched by calling stk.pop() to get that location. in this process if an exception is encountered then this means that the ')' is not matched so print the message accordingly.
8. End **for** loop
9. Start **while** loop with condition *!stk.isEmpty()*
10. Print the location where '(' has been matched by calling the method stk.pop()
11. End **while** loop

**import** **java.util.\***;

**class** Node {

**protected** **int** data;

**protected** Node next = **null**;

**public** Node() {

}

**public** Node(**int** data, Node next) {

**this**.data = data;

**this**.next = next;

}

**void** setData(**int** data) {

**this**.data = data;

}

**void** setLink(Node link) {

**this**.next = link;

}

Node getLink() {

**return** next;

}

**int** getData() {

**return** data;

}

}

**class** Stack {

**protected** Node top = **null**;

**public** **void** push(**int** val) {

*// Currently we have set no limits to pushing elements*

Node nptr = **new** Node(val, top);

top = nptr;

}

**public** **int** pop() **throws** Exception {

**if** (isEmpty()) {

**throw** **new** Exception("Stack Underflow");

} **else** {

**int** val = top.getData();

top = top.getLink();

**return** val;

}

}

**public** **boolean** isEmpty() {

**return** top == **null**;

}

**public** **void** show() {

Node ptr = top;

**while** (ptr != **null**) {

System.out.printf("%d, ", ptr.getData());

ptr = ptr.getLink();

}

}

}

**public** **class** MatchParenStack {

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

Scanner sc = **new** Scanner(System.in);

System.out.printf("Enter string to Match Parenthesis:\n");

String val = sc.nextLine();

Stack stk = **new** Stack();

**for** (**int** x = 0; x < val.length(); x++) {

**char** ch = val.charAt(x);

**if** (ch == '(')

stk.push(x);

**else** **if** (ch == ')')

**try** {

**int** ind = stk.pop();

System.out.printf("The ) at index %d matched with ( at index %d\n", x, ind);

} **catch** (Exception e) {

System.out.printf("The ) at index %d is UNMATCHED!!\n", x);

}

}

**while** (!stk.isEmpty()) {

**try** {

System.out.printf("The ( at index %d is UNMATCHED!!\n", stk.pop());

} **catch** (Exception e) {

}

}

sc.close();

}

}

## Variable Description

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **Class Node** |  |  |
| **global** |  |  |
| data | int | value to be stored in the current node |
| link | Node | pointer for the next node |
| **Class Stack** |  |  |
| **global** |  |  |
| top | Node | topmost element in the stack |
| **public void push(int val)** |  |  |
| nptr | Node | to store the new pointer which will eventually become the top |
| **public int pop()** |  |  |
| val | int | to temporarily store data of the top most element to be returned |
| **public void show()** |  |  |
| ptr | Node | a pointer to be used in the loop to traverse in the stack |
| **Class MatchParenStack** |  |  |
| **void main()** |  |  |
| sc | Scanner | object to accept user input |
| val | String | to store entered expression to evaluate |
| stk | Stack | a stack data type object to help us in parenthesis matching |
| x | int | counter variable to iterate over 'val' |
| ch | char | to store current extracted character |

**Program 25**

**Infix Expression to Postfix Expression Converter**.

Example:- l\*(p+f\*(d+g)) -> lpfdg+\*+\*

## Algorithm

**Class Node**

#### public Node(char d, Node next)

1. data=d, data
2. link=next, to store the link to the next node

#### public void setLink(Node n)

1. link=n, set the link to the next node

#### public void setData(char d)

1. data=d, set the value of current Node

#### public int getData()

1. return data(value of the current node)

#### public Node getLink()

1. return link

**Class Stack**

#### public void push(char val)

1. Create a Node nptr with value of 'val' and 'top' as its link(next element)
2. top=nptr

#### public char pop()

1. If isEmpty() returns true then throw a Exception with message "Stack Underflow"
2. Start **else**
3. Store val=top.getData()
4. top=top.getLink()
5. return val
6. End **else**

#### public boolean isEmpty()

1. if top==null then return true else return false

#### public void show()

1. Initailize a Node ptr=top
2. Start **while** loop with condition ptr!=null
3. Print ptr.getData()
4. ptr=ptr.getLink() (going to the next element in the stack)
5. End **while** loop

**Class Convert**

#### public Convert(String str)

1. data = str.toCharArray()

#### String getPostfix()

1. Initialize a String res to store the final output
2. Initaialize a Stack 'stk'
3. Store first\_op=true
4. Store bracket=false
5. Start **for** loop with x=0, x<data.length, x=x+1
6. Store character ch=data[x]
7. Store integer prec=isBodmasOperrator(ch)
8. Start **if** with prec==-1
9. Start **if** with ch=='('
10. first\_op=true
11. bracket=true
12. stk.push(ch)
13. End **if**
14. Start **else if** with ch==')'
15. Start **for** loop with op=stk.pop(), op!='(' and op=stk.pop()
16. res=res+ch
17. End **for** loop
18. If while executing **this** block of code exception ocuurs then throw a Exception with message "Invalid Expression"
19. End **if else**
20. **else** res=res+ch
21. End **if**
22. Start **if else** with condition prec!=-1 (i.e. a BEDMAS operator is detected)
23. Start **if** with prv>=prec and !first\_op
24. Start **while** loop with condition !stk.isEmpty()
25. Store character vl=stk.pop()
26. if vl=="(" and bracket is true then stk.push(vl) and **break**
27. res=res+vl
28. End **while** loop
29. End **if**
30. stk.push(ch)
31. prv=prec
32. first\_op=false
33. End **if else**
34. End **for** loop
35. Start **while** loop with !stk.isEmpty()
36. Store vl=stk.pop()
37. if vl=='(' and bracket is true then throw an Eception with message "Invalid Expression"
38. res=res+vl
39. End **while**
40. return res

#### private int isBeadmasOperator(char ch)

1. Start **for** loop with x=0, x<BEDMAS.length and x=x+1
2. if BEDMAS[x] == ch then retuen BEDMAS\_PREC[x]
3. End **for** loop
4. return -1, as no operator detected.

**Class InfixtoPostfix**

#### void main()

1. Initialize the Scanner object to accept user input
2. Input the Infix Expression in 'val'
3. Initialize a object 'obj' of class Convert with 'val' as parameter
4. Print the converted Postfix Expression by using obj.getPostfix() if error is encounterd print the error message

**import** **java.util.\***;

**class** Node {

**protected** **char** data;

**protected** Node next = **null**;

**public** Node() {

}

**public** Node(**char** data, Node next) {

**this**.data = data;

**this**.next = next;

}

**void** setData(**char** data) {

**this**.data = data;

}

**void** setLink(Node link) {

**this**.next = link;

}

Node getLink() {

**return** next;

}

**char** getData() {

**return** data;

}

}

**class** Stack {

**protected** Node top = **null**;

**public** **void** push(**char** val) {

*// Currently we have set no limits to pushing elements*

Node nptr = **new** Node(val, top);

top = nptr;

}

**public** **char** pop() **throws** Exception {

**if** (isEmpty()) {

**throw** **new** Exception("Stack Underflow");

} **else** {

**char** val = top.getData();

top = top.getLink();

**return** val;

}

}

**public** **boolean** isEmpty() {

**return** top == **null**;

}

}

**class** Convert {

**private** **char**[] BEDMAS = { '-', '+', '\*', '/', '^' };

**private** **int**[] BEDMAS\_PREC = { 0, 0, 1, 1, 2 };

*// Not Exactly follwing the BEADMAS rule here, but going with operator*

*// precedence rules*

*// To use BEDMAS to convert don't use BEDMAS\_PREC and directly get the index*

*// from BEDMAS*

**private** **char**[] data;

**public** Convert(String str) {

data = str.toCharArray();

}

**public** String getPostfix() **throws** Exception {

String res = "";

Stack stk = **new** Stack();

**int** prv = -1; *// storing previous precedence for comparision*

**boolean** first\_op = **true**;

**boolean** bracket = **false**;

**for** (**int** x = 0; x < data.length; x++) {

**char** ch = data[x];

**int** prec = isBeadmasOperator(ch);

**if** (prec == -1) {

**if** (ch == '(') {

first\_op = **true**;

bracket = **true**;

stk.push(ch);

} **else** **if** (ch == ')')

**try** {

**for** (**char** op = stk.pop(); op != '('; op = stk.pop())

res += op;

bracket = **false**;

} **catch** (Exception e) {

**throw** **new** Exception("Invalid Expression");

}

**else**

res += ch;

} **else** **if** (prec != -1) { *// BEDMAS operaator detected*

**if** (prv >= prec && !first\_op)

**while** (!stk.isEmpty()) {

**char** vl = stk.pop();

**if** (vl == '(' && bracket) {

stk.push(vl);

**break**;

}

res += vl;

}

stk.push(ch);

prv = prec;

first\_op = **false**;

}

}

**while** (!stk.isEmpty()) {

**char** vl = stk.pop();

**if** (vl == '(' && bracket)

**throw** **new** Exception("Invalid Expression");

res += vl;

}

**return** res;

}

**private** **int** isBeadmasOperator(**char** ch) {

**for** (**int** x = 0; x < BEDMAS.length; x++)

**if** (BEDMAS[x] == ch)

**return** BEDMAS\_PREC[x];

**return** -1;

}

}

**public** **class** InfixToPostfix {

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

Scanner sc = **new** Scanner(System.in);

System.out.println("Enter Infix Expression:");

String val = sc.nextLine();

Convert obj = **new** Convert(val);

**try** {

System.out.printf("Converted Postfix Expression:\n%s\n", obj.getPostfix());

} **catch** (Exception e) {

System.out.println(e.getMessage());

}

sc.close();

}

}

**Variable Description**

| **Name** | **Type** | **Uses** |
| --- | --- | --- |
| **Class Node** |  |  |
| ***global*** |  |  |
| data | char | value to be stored in the current node |
| link | Node | pointer for the next node |
| **Class Stack** |  |  |
| ***global*** |  |  |
| top | Node | topmost element in the stack |
| ***public void push(int val)*** |  |  |
| nptr | Node | to store the new pointer which will eventually become the top |
| ***public int pop()*** |  |  |
| val | char | to temporarily store data of the top most element to be returned |
| ***public void show()*** |  |  |
| ptr | Node | a pointer to be used in the loop to traverse in the stack |
| **Class Convert** |  |  |
| ***global*** |  |  |
| BEDMAS | char[] | BEDMAS Oprtaor i.e. -,+,\*,/,^ |
| BEDMAS\_PREC | int[] | the precedence of the 'BEDMAS' operator |
| data | char[] | to store the entered expression as character array |
| ***public String getPostfix()*** |  |  |
| res | String | to store resultant postfix expresion |
| prv | int | storing the precedence value of previous encountered operator |
| first\_op | boolean | indicates start of the expression or '(' bracket |
| bracket | boolean | status if the current index is index a bracket |
| ch | char | to store the character at the current index |
| prec | int | precision value of the current character |