**Computer Science Project**

Aanjishnu Bhattacharyya

10th January 2021

**Assignment 11**

Class name: Change

Data Members/instance variables:

str: stores the word

newstr: stores the changed word

len: store the length of the word

Member functions:

Change(): default constructor

void inputword( ): to accept a word

char caseconvert (char ch): converts the case of the character and returns it

void recchange (int): extracts characters using recursive technique and changes its case using caseconvert () and forms a new word

void display (): displays both the words

Specify the class Change, giving details of the Constructor ( ), member functions void inputword (), char caseconvert (char ch), void recchange (int) and void display (). Define the main () function to create an object and call the functions accordingly to enable the above change in the given word.

**Alogorithm:**

Class Change:

Method inputword:

Step 1: create Scanner method

Step 2: input a word into str

Step 3: set the value of len to str’s length

Method caseconvert:

Step 1: if the argument is lower case letter return the corrosponding capital letter

Step 2: if the argument is upper case letter return the corrosponding small letter

Step 3: otherwise just return the argument

Method recchange:

Step 1: if arg is greater than len just return

Step 2: append caseconvert(str[a]) to newstr

Step 3: call recchange(a + 1)

Method display:

Step 1: print str

Step 2: print newline

Step 3: print newstr

Class Change\_main:

Method main:

Step 1: create a Change object

Step 2: print Enter str:

Step 3: call inputword

Step 4: call recchange

Step 5: call displays

**Source code:**

import java.util.Scanner ;

public class Change

{

private String str; // stores the word

private String newstr; // stores the changed word

private int len; // length of the word

Change()

{

this.str = "";

this.newstr = "";

this.len = 0;

}

// input a single word

void inputword()

{

Scanner sc = new Scanner(System.in);

this.str = sc.next();

this.len = this.str.length();

}

// change the case of the letter

char caseconvert(char ch)

{

if(ch >= 'a' && ch <= 'z') return (char)(ch - 'a' + 'A');

if(ch >= 'A' && ch <= 'Z') return (char)(ch - 'A' + 'a');

return ch;

}

void recchange(int a)

{

if(a >= this.len) return;

this.newstr += caseconvert(this.str.charAt(a));

recchange(a+1);

}

// display the strings

void display()

{

System.out.println(this.str+"\n"+this.newstr);

}

}

public class Change\_main

{

// entrypoint very cool

public static void main(String args[])

{

Change c = new Change();

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

c.inputword();

c.recchange(0);

c.display();

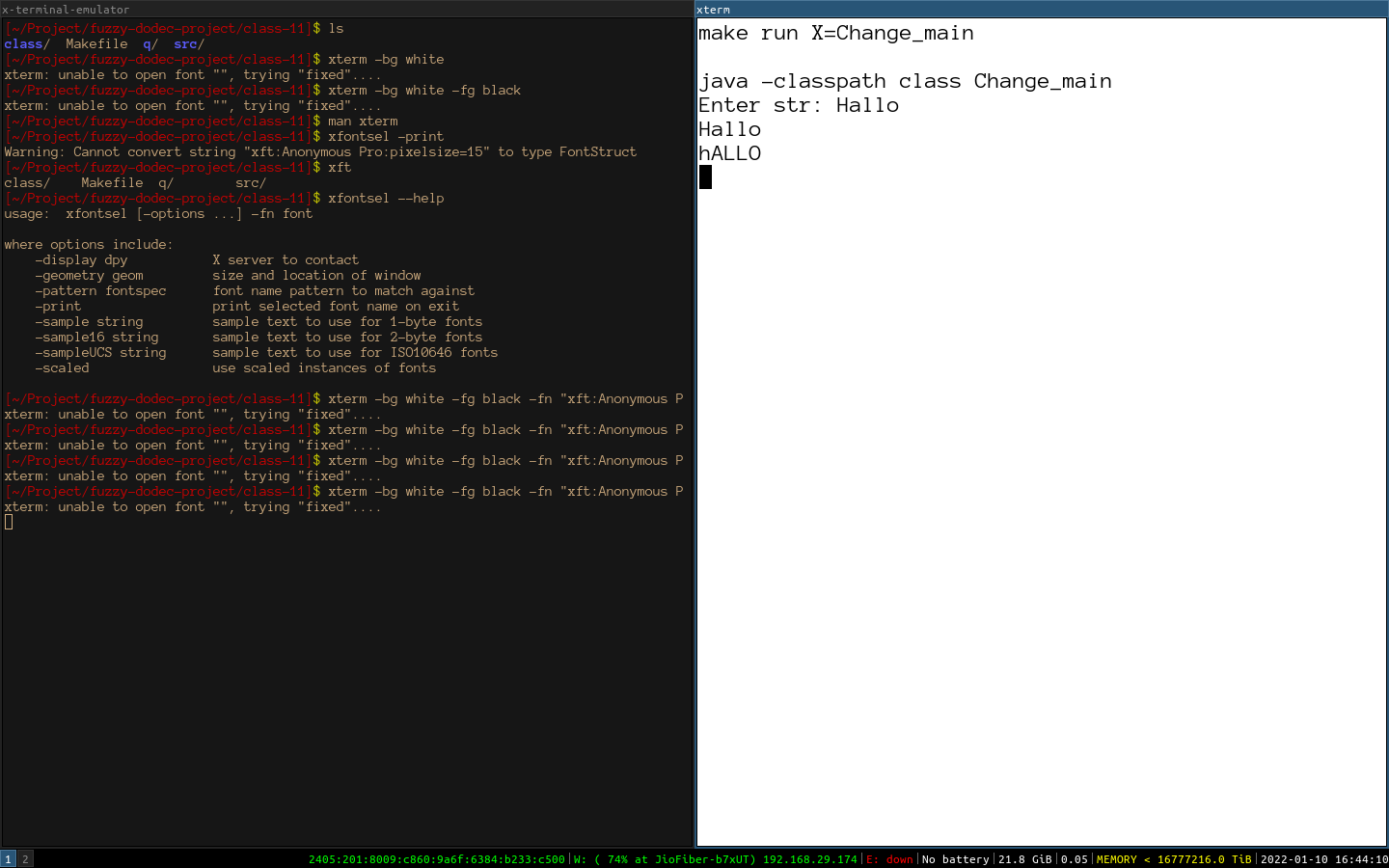
}

}

**Variable Listing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Function** | **Data Type** | **Scope** |
| str | Stores the word | String | Change |
| newstr | Stores the changed word | String | Change |
| len | Length of the word | int | Change |
| ch | Change the case of letter | char | caseconvert |
| a | Index of the the character | int | recchange |
| sc | Input handle | Scanner | inputword |

**Output:**



**Assignment 12**

A class DeciOct has been defined to convert a decimal number into its equivalent octal number. Some of the members of the class are given below:

Class name: DeciOct

Data members/instance variables:

n: stores the decimal number

oct: stores the octal equivalent number

Member functions:

DeciOct(): constructor to initialize the data members n = 0, oct = 0.

void getnum(int nn): assign nn to n

void deci\_oct(): calculates the octal equivalent of ‘n’ and stores it in oct using the recursive technique void show(): displays the decimal number ‘n’, calls the function deci\_oct() and displays its octal equivalent.

Specify the class DeciOct, giving details of the constructor( ), void getnum(int), void deci\_oct( ) and void show(). Also define a main() function to create an object and call the functions accordingly to enable the task.

**Algorithm**:

Class DeciOct:

Method getnum:

Step 1: set the value of N to n

Method deci\_oct:

Step 1: if n <= 0 then

Step 2: let x to be oct

Step 3: while x is not zero

Step 4: let oct = oct \* 10 + x % 10

Step 5: x devide equals 10

Step 6: otherwise do

Step 7: let oct = oct \* 10 + n % 8

Step 8: let n devide equals 8

Step 9: call deci\_oct

Method show:

Step 1: print original num

Step 2: call deci\_oct

Step 3: print octal number

Class DeciOct\_main:

Method main:

Step 1: create objects

Step 2: print Enter a number:

Step 3: get the number using scanner and then send it to getnum

Step 4: call show method

**Source code:**

public class DeciOct

{

private int n; // stores the decimal number

private int oct; // stores the octal equivalent number

// construct to initialize the data members

DeciOct()

{

this.n = 0;

this.oct = 0;

}

// assign nn to n

void getnum(int nn)

{

this.n = nn ;

}

// change to octal

void deci\_oct()

{

if(this.n <= 0)

{

int x = this.oct ;

this.oct = 0 ;

while(x != 0)

{

this.oct = this.oct \* 10 + x % 10 ;

x /= 10 ;

}

}

else

{

this.oct = this.oct \* 10 + this.n % 8 ;

this.n /= 8 ;

deci\_oct() ;

}

}

// display the decimal and octal number

void show()

{

System.out.println(this.n);

deci\_oct();

System.out.println(this.oct);

}

}

import java.util.Scanner ;

public class DeciOct\_main

{

public static void main(String args[])

{

DeciOct d = new DeciOct() ;

Scanner sc = new Scanner(System.in) ;

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

d.getnum(sc.nextInt()) ;

d.show() ;

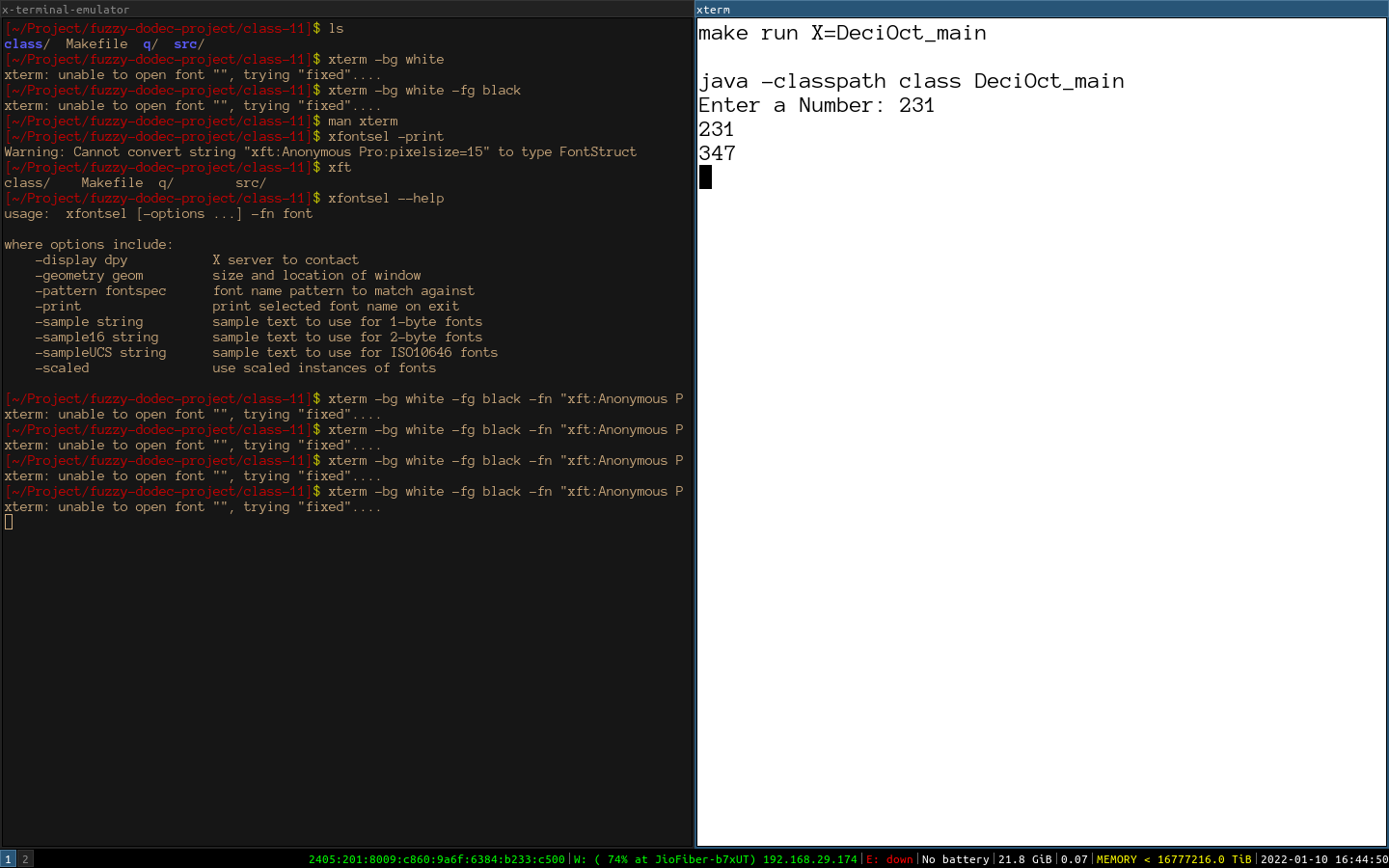
}

}

**Variable Listing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Function** | **Data Type** | **Scope** |
| n | Stores the decimal number | int | DeciOct |
| oct | Octal number of system | int | DeciOct |
| nn | Temporary input value | int | getnum |
| x | Temporary variable for  reversal | int | deci\_oct |

**Output:**



**Assignment 13**

Design a class Exchange to accept a sentence and interchange the first alphabet with the last alphabet for each word in the sentence, with single-letter word remaining unchanged. The words in the input sentence are separated by a single blank space and terminated by a full stop.

Example:

Input: It is a warm day.

Output: tIsi a mraw yad

Some of the data members and member functions are given below:

Class name: Exchange

Data members/instance variables:

sent: stores the sentence

rev: to store the new sentence

size: stores the length of the sentence

Member functions:

Exchange(): default constructor

void readsentence(): to accept the sentence

void exfirstlast(): extract each word and interchange the first and last alphabet of the word and form a new sentence rev using the changed words.

void display(): display the original sentence along with the new changed sentence.

Specify the class Exchange giving details of the constructor ( ), void readsentence (), void exfirstlast () and void display (). Define the main () function to create an object and call the functions accordingly to enable the task.

**Algorithm:**

Class Exchange:

Method readsentence:

Step 1: create a scanner object

Step 2: read a line of input from standard input

Method exfirstlast:

Step 1: create a array called f by using calling the split(“ “) method on sentence

Step 2: using g and iterate over everything in f

Step 3: use a temporary variable called t

Step 4: loop form g.length()-1 to 0 using i

Step 5: at char at I of g to t

Step 6: append the temporary t to the end of reversal

Method display:

Step 1: print sent

Step 2: print a newline

Step 3: print rev

Class Exchange\_main:

Method main:

Step 1: create new Exchange object

Step 2: call readsentence

Step 3: call exfirstlast

Step 4: call displays

**Source code:**

import java.util.Scanner;

public class Exchange

{

private String sent; // stores the sentence

private String rev; // to store the new sentence

private int size; // stores the length of the sentence

Exchange()

{

this.sent = "";

this.rev = "";

this.size = 0;

}

// to accept sentence

void readsentence()

{

Scanner sc = new Scanner(System.in);

this.sent = sc.nextLine() ;

}

// extract each word and

void exfirstlast()

{

String f[] = this.sent.split(" ") ;

for(String g : f)

{

String t = "" ;

for(int i = g.length()-1; i >= 0; i--)

{

t += g.charAt(i) ;

}

this.rev += t + " " ;

}

}

void display()

{

System.out.println(this.sent+"\n"+this.rev) ;

}

}

public class Exchange\_main

{

public static void main(String args[])

{

Exchange e = new Exchange() ;

e.readsentence() ;

e.exfirstlast() ;

e.display() ;

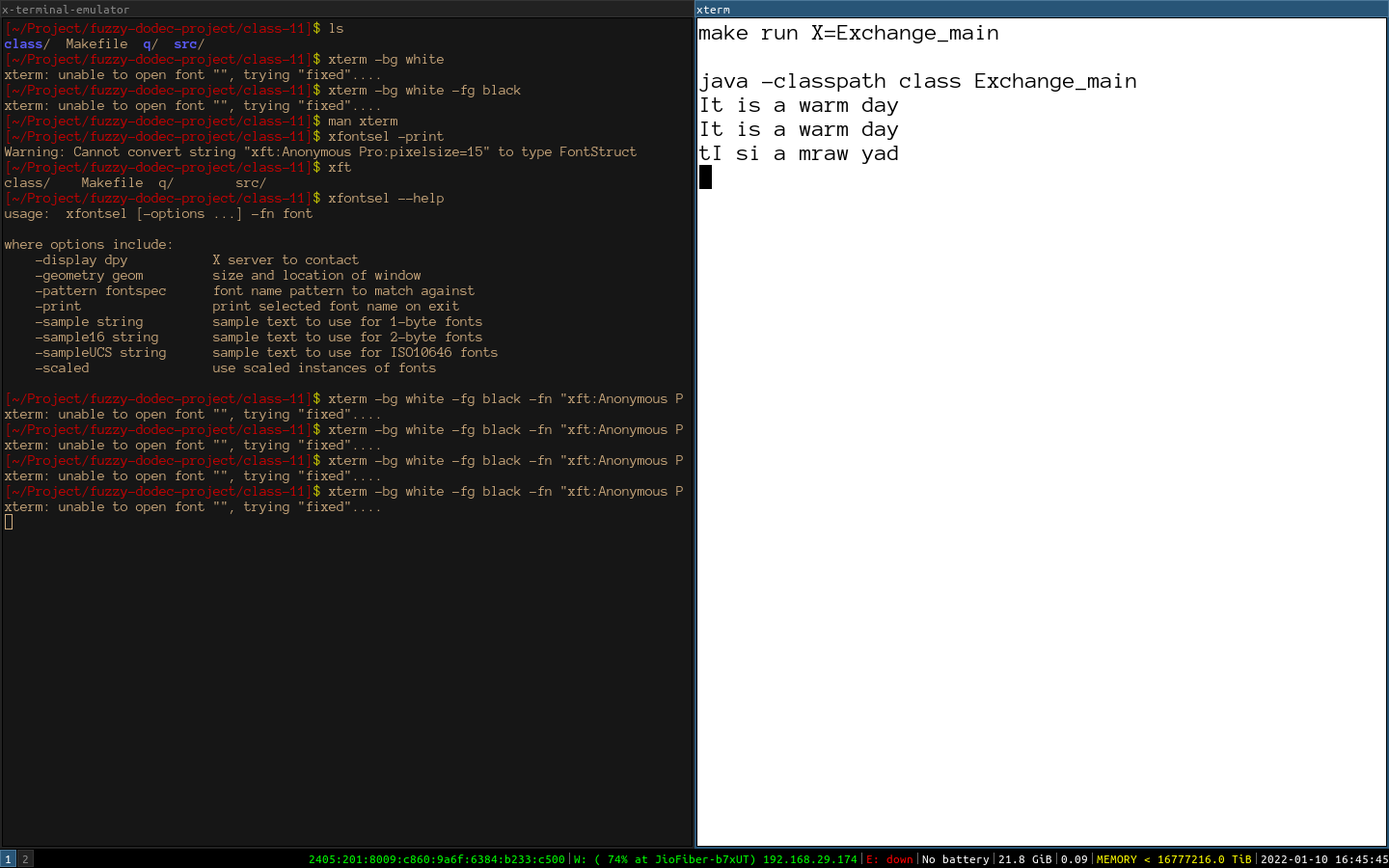
}

}

**Variable Listing:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Function | Data Type | Scope |
| sent | Stores the sentences | String | Exchange |
| rev | To store the new sentence | String | Exchange |
| size | Stores the length of the sentences | String | Exchange |
| g | Iterator static variable | String | exfirstlast |
| f | Array that holds the list of words | String[] | exfirstlast |
| t | Temporary reversed word | String | exfirstlast |
| i | The location of the char on the list | int | exfirstlast |
| sc | A handle to accept input | Scanner | readsentence |

**Output:**



**Assignment 14**

Write a program which takes a string (maximum 80 characters terminated by a full stop. The words in this string are assumed to be separated by one or more blanks. Arrange the words of the input string in descending order of their lengths. Same length words should be sorted alphabetically. Each word must start with an uppercase letter and the sentence should be terminated by a full stop.

Test your program for the following data and some random data.

SAMPLE DATA:

INPUT: "This is human resource department."

OUTPUT: Department Resource Human This Is.

INPUT: "To handle yourself use your head and to handle others use your heart."

OUTPUT: Yourself Handle Handle Others Heart Head Your YourAnd Use Use To To.

**Algorithm:**

Class Arrange:

Method arrange:

Step 1: create a variable called words. Array filled with words.

Step 2: bubble sort the words ind decending order depending on length and alphbetic order

Step 3: create a sentence using the sorted words

Step 4: conver upper case the first char arr and add . At the end

Method display:

Step 1: print sent

Step 2: print arr

Class Arrange\_main:

Method main:

Step 1: create input handler

Step 2: accept string input

Step 3: check if the ending of string is a ‘.’ and it is < 80 chars long and it has a upper case start

Step 4: call arrange

Step 5: call displays

**Source Code:**

public class Arrange

{

private String sent ;

private String arr ;

Arrange(String s)

{

this.sent = s ;

this.arr = "" ;

}

void arrange()

{

String words[] = this.sent.toLowerCase().substring(0, this.sent.length()-1).split(" ") ;

for(int i = 0; i < words.length; i++)

{

for(int j = 0; j < words.length - 1; j++)

{

if(words[j].length() < words[j+1].length())

{

String k = words[j] ;

words[j] = words[j+1] ;

words[j+1] = k ;

}

else if(words[j].length() == words[j+1].length())

{

if(words[j].compareTo(words[j+1]) < 0)

{

String k = words[j] ;

words[j] = words[j+1] ;

words[j+1] = k ;

}

}

}

}

for(int i = 0; i < words.length ; i++)

{

this.arr += words[i] ;

if(i != words.length - 1)

this.arr += " " ;

}

this.arr = Character.toUpperCase(this.arr.charAt(0)) + this.arr.substring(1) + "." ;

}

void display()

{

System.out.println(this.sent) ;

System.out.println(this.arr) ;

}

}

import java.util.Scanner ;

public class Arrange\_main

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

String s = sc.nextLine() ;

if(s.length() > 80 ||

Character.isLowerCase(s.charAt(0)) ||

s.charAt(s.length()-1) != '.')

{

System.out.println("Wrong input");

return;

}

Arrange a = new Arrange(s);

a.arrange();

a.display();

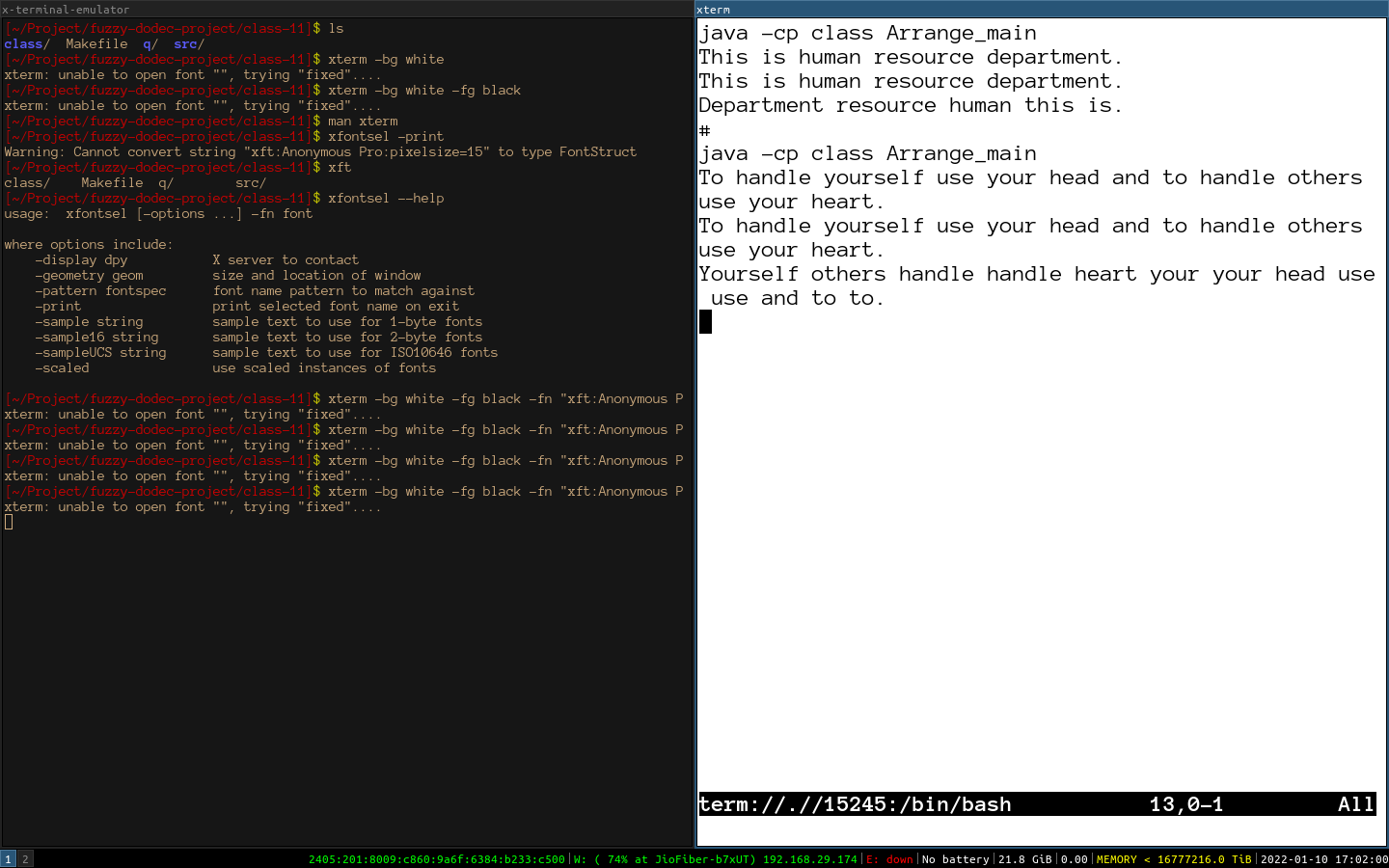
}

}

**Variable Listing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Function** | **Data Type** | **Scope** |
| sent | Sentence storage | String | Arrange |
| arr | Output sentence | String | Arrange |
| i | Iterative sentence | int | arrange |
| j | Iterative number | int | arrange |
| k | Temporary variable | String | arrange |

**Output:**



**Assignment 15**

A magic square of order n is an arrangement of n^2 numbers, usually distinct integers, in a square, such that the n numbers in all rows, all columns, and both diagonals sum to the same constant. A magic square contains the integers from 1 to n^2. Value of n must be considered as an odd number.

**Algorithm:**

Class Magic:

Method gensq:

Step 1: set row to be n -1

Step 2: set col to be n/2

Step 3: magic [row][col] = 1

Step 4: loop from 2 to n squared with one increment

Step 5: if the position (row + 1) % n and (col + 1) % n is zero

Step 6: then set row to be (row + 1) and col to be (col+1) modulated by n

Step 7: else only set row to be (row – 1 + n) modulated by n

Step 6: set magic at [row][col] be I

Method display:

Step 1: outer loop of I from 0 to n

Step 2: inner loop of j from 0 to n

Step 3: print some adjustment spaces

Step 4: display magic[i][j]

Class Magic\_main:

Method main:

Step 1: check for constrainsts

Step 2: take intpu

Step 3: call gensq

Step 4: call displays

**Source Code:**

import java.util.Scanner ;

public class Magic

{

private int magic[][] ;

private int n ;

Magic(int N)

{

this.n = N ;

this.magic = new int[N][N] ;

}

void gensq()

{

int row = n-1;

int col = n/2;

magic[row][col] = 1;

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

{

if (magic[(row + 1) % n][(col + 1) % n] == 0)

{

row = (row + 1) % n;

col = (col + 1) % n;

}

else

{

row = (row - 1 + n) % n;

}

magic[row][col] = i;

}

}

void display()

{

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

{

for (int j = 0; j < n; j++)

{

if (magic[i][j] < 10) System.out.print(" ");

if (magic[i][j] < 100) System.out.print(" ");

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

}

System.out.println();

}

}

}

import java.util.Scanner ;

public class Magic\_main

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.print("N= ") ;

int N = sc.nextInt() ;

if(N % 2 == 0)

{

System.out.println("INVALID INPUT") ;

return ;

}

Magic m = new Magic(N) ;

m.gensq() ;

m.display() ;

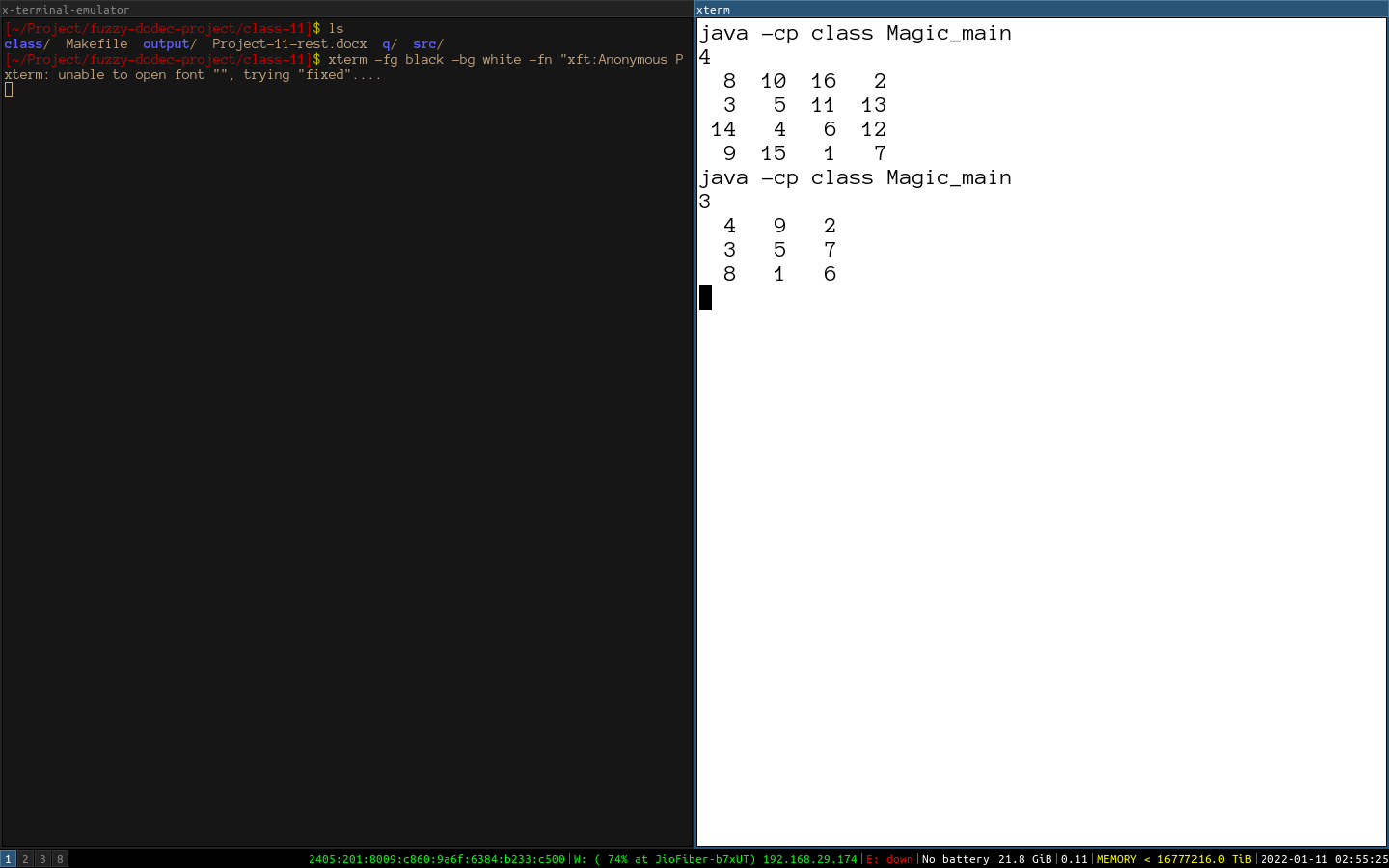
}

}

**Variable Listing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Function** | **Data Type** | **Scope** |
| row | Row number in buffer | int | gensq |
| col | Col number in buffer | int | gensq |
| i | Iterator variable | int | gensq |
| j | Iterator variable | int | gensq |
| magic | The magic matrix actual | int[][] | Magic |
| n | The side of matrix | int | Magic |

**Output:**

****

**Assignment 16**

A special pure number is a number if a) It has even number of digits b) It contains digits 4 or 5 c) It is a palindrome number. Write a program to accept a positive number N from the user and print first Nth positive special pure

**Algorithm:**

Class SpecialPureNumber:

Method isSPN:

Step 1: check if the number has even number or digits if not return false

Step 2: reverse the number using a while loop and a duplicate of the original

Step 3: check if all the digits in the numbers are either 4 or 5 if not then return flase

Step 4: check if the reverse is equal to original if not then return false

Step 5: return true if all else fails.

Class SpecialPureNumber\_main

Method main:

Step 1: accept a number less than 10

Step 2: start from 10 then increment forward checking if the numbers are SPN

Step 3: if n SPN’s are found then end the loop

Step 4: print the SPN’s

**Source Code:**

public class SpecialPureNumber

{

boolean isSPN(int x)

{

// check if the number of digits in x is even or not

if((int)(Math.log10(x) + 1) % 2 != 0) return false;

// loop through all the digits of the number and make sure they

// are all 4s or 5s

int d = x ;

int rev = 0 ; // why waste a fairly good loop lets also find the reverse of x

while(d != 0)

{

if(d % 10 != 4 && d % 10 != 5) return false ;

rev = rev \* 10 + d % 10 ;

d /= 10 ;

}

if(rev != x) return false;

return true ;

}

}

import java.util.Scanner;

public class SpecialPureNumber\_main

{

public static void main(String args[])

{

SpecialPureNumber s = new SpecialPureNumber() ;

Scanner sc = new Scanner(System.in) ;

System.out.print("N= ") ;

int n = sc.nextInt() ;

if(n > 10 || n <= 0)

{

System.out.println("OUT OF RANGE") ;

return ;

}

int i = 10 ;

while(n != 0)

{

if(s.isSPN(i))

{

System.out.print(i+" ") ;

n-- ;

}

i++ ;

}

System.out.println() ;

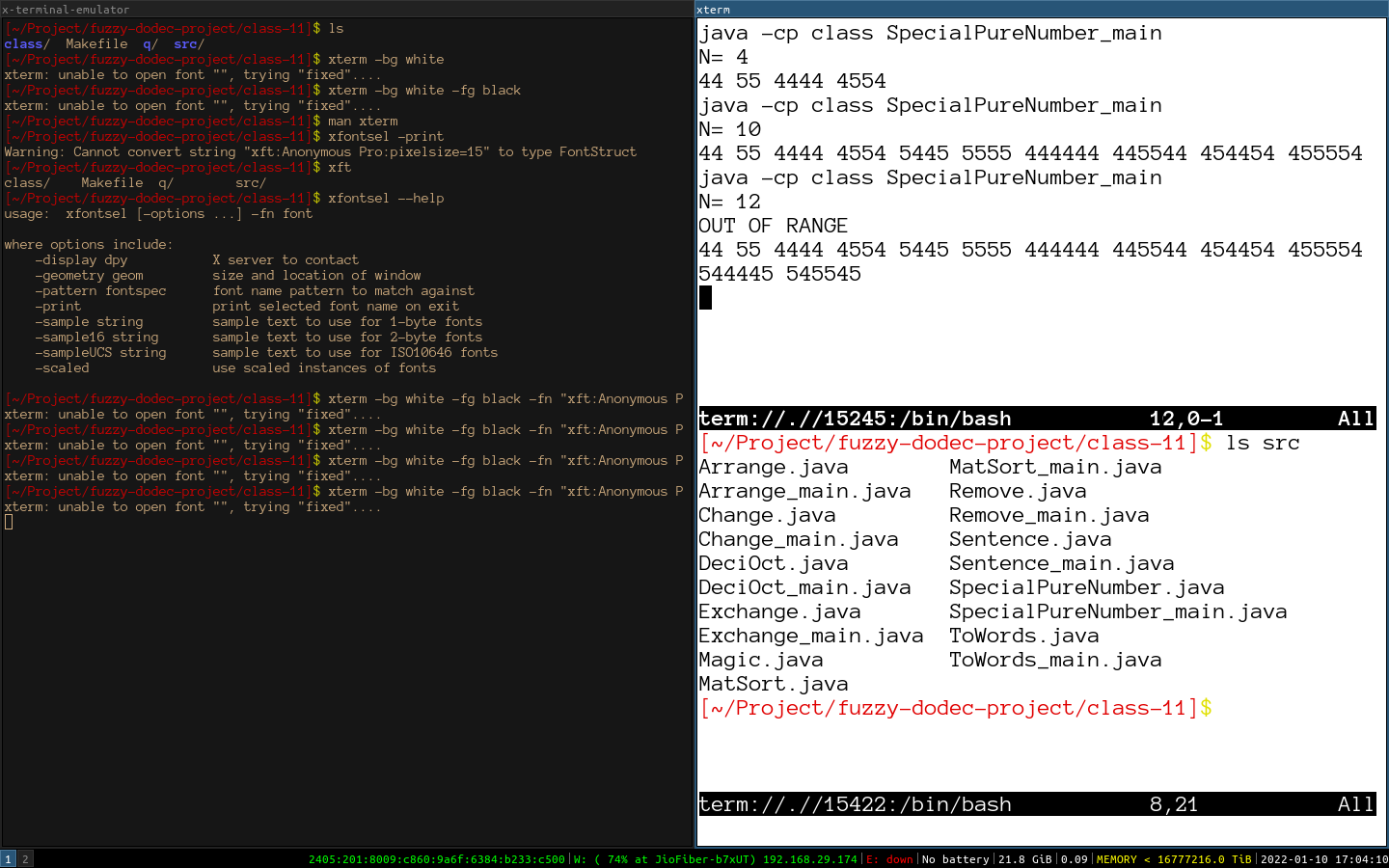
}

}

**Variable Listing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Function** | **Data Type** | **Scope** |
| d | Duplicate of number x | int | isSPN |
| rev | The revers of x | int | isSPN |
| x | Number input of the special input number check | int | isSPN |
| n | Number of numbers required | int | main |

**Output:**



**Assignment 17**

Write a program to declare a square matrix A[ ][ ] of order (m) where 'm' is the number of rows which be greater than 3 and less than 20. Allow the user to input positive integers into this matrix. Perform the following tasks on the matrix: (a) Print the original matrix. (b) Sort only the BOUNDARY elements in descending order using anystandard sorting tech

**Algorithm:**

Class MatSort:

Method boundarySort:

Step 1: create a buffer called x

Step 2: copy the boundary of matrix on x

Step 3: sort x using bubble sort

Step 4: copy x back to the boundary of the matrix

Method digdisp:

Step 1: outer loop from i to N

Step 2: inner loop from j to N

Step 3: print if i==j or if i==N-j-1 then print A[i][j]

Step 4: else print ‘ ’

Step 5: print newline

Method display:

Step 1: outer loop from i to N

Step 2: inner loop from j to N

Step 3: print A[i][j]

Step 4: print newline

Method setA:

Step 1: set A[i][j] to x

Method getN:

Step 1: return N

Class MatSort\_main:

Method main:

Step 1: create input handler

Step 2: take input of size N

Step 3: check if N is bigger than 3 and less than 20

Step 4: fill up the Matrix

Step 5: display accroding to requirement

**Source Code:**

public class MatSort

{

private int A[][] ;

private int N ;

MatSort(int N)

{

this.N = N ;

this.A = new int[N][N] ;

}

void boundarySort()

{

int x[] = new int[4 \* (N-1)] ;

int ix = 0 ;

for(int i = 0; i < N-1; i++) x[ix++] = A[0][i];

for(int i = 0; i < N-1; i++) x[ix++] = A[i][N-1];

for(int i = N-1; i >= 1; i--) x[ix++] = A[N-1][i];

for(int i = N-1; i >= 1; i--) x[ix++] = A[i][0];

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

{

for(int j = 0; j < x.length - 1; j++)

{

if(x[j] < x[j+1])

{

int t = x[j] ;

x[j] = x[j+1] ;

x[j+1] = t ;

}

}

}

ix = 0 ;

for(int i = 0; i < N-1; i++) A[0][i] = x[ix++];

for(int i = 0; i < N-1; i++) A[i][N-1]= x[ix++];

for(int i = N-1; i >= 1; i--) A[N-1][i]= x[ix++];

for(int i = N-1; i >= 1; i--) A[i][0] = x[ix++];

}

void digdisp()

{

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

{

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

{

if(i == j || i == N-j-1)

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

else

System.out.print(" ") ;

}

System.out.print('\n') ;

}

}

void display()

{

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

{

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

{

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

}

System.out.print('\n') ;

}

}

public void setA(int i, int j, int x)

{

A[i][j] = x ;

}

public int getN()

{

return N ;

}

}

import java.util.Scanner ;

public class MatSort\_main

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in) ;

System.out.print("M= ") ;

MatSort x = new MatSort(sc.nextInt()) ;

if(x.getN() > 20 || x.getN() < 3)

{

System.out.println("OUTRANGE") ;

return ;

}

for(int i = 0; i < x.getN(); i++)

{

for(int j = 0; j < x.getN(); j++)

{

x.setA(i, j, sc.nextInt());

}

}

System.out.println("ORIGINAL MATRIX: ") ;

x.display() ;

x.boundarySort() ;

System.out.println("REARRANGED MATRIX: ") ;

x.display() ;

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

x.digdisp() ;

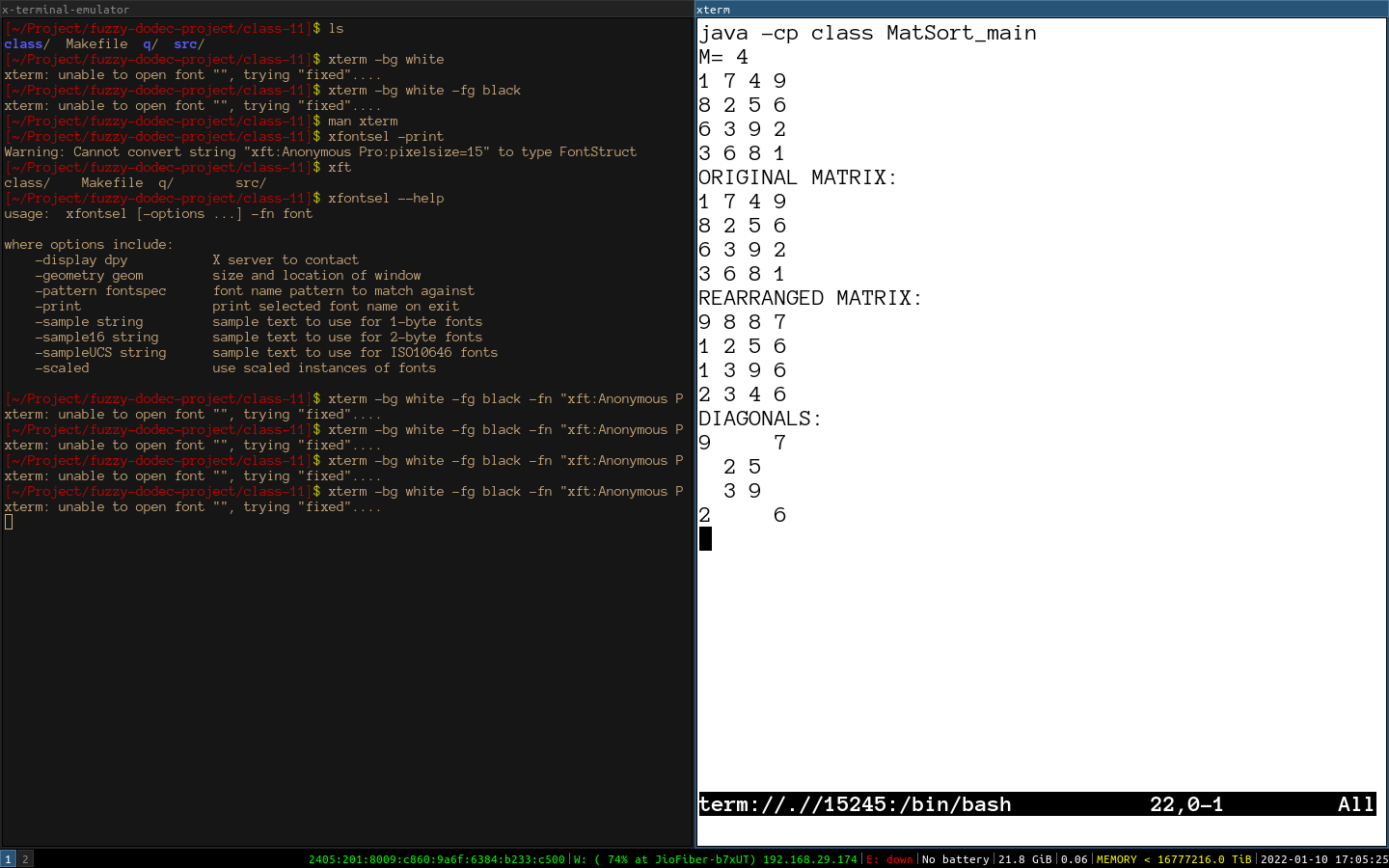
}

}

**Variable listing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Function** | **Data Type** | **Scope** |
| N | Size of the matrix | int | MatSort |
| A | Matrix that stores the things | int[][] | MatSort |
| i | Index or row number location | int | boundarySort |
| j | Index or column number location | int | boundarySort |
| ix | Index of x as a linear location | int | boundarySort |
| t | Temporary variable to save value | int | boundarySort |

**Output:**



**Assignment 18**

Write a program which will accept a positive natural number from the user which will not exceed four digits. The program will print the display the number in words as shown in the example. The program will be repeated for as many times the user wants

**Algorithm:**

Class ToWords:

Method towords:

Step 1: collect the face values of the number provided

Step 2: find the relevant words from the look up tables

Step 3: special case for ‘teens’

Step 4: return the string word that is produced

Class ToWords\_main:

Method main:

Step 1: create input handler

Step 2: take input of number to be stringified

Step 3: check for the constraints

Step 4: print the output after calling towords

Step 5: loop until n is provided

**Source Code:**

public class ToWords

{

private final String ones[] = {"one", "two", "three", "four", "five", "six", "seven", "eight", "nine"} ;

private final String teen[] = {"ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"} ;

private final String tens[] = {"", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety"} ;

String towords(int x)

{

String out = "" ;

int d = (x / 1000) % 10 ;

int h = (x / 100) % 10 ;

int t = (x / 10) % 10 ;

int o = (x / 1) % 10 ;

if(d != 0) out += ones[d-1] + " thousand " ;

if(h != 0) out += ones[h-1] + " hundred and " ;

if(t != 0)

{

if(t == 1)

{

out += teen[t\*10 + o - 10] ;

o = 0 ;

}

else out += tens[t-1] + " " ;

}

if(o != 0) out += ones[o - 1] ;

return out ;

}

}

import java.util.Scanner ;

public class ToWords\_main

{

public static void main(String args[])

{

ToWords t = new ToWords() ;

Scanner sc = new Scanner(System.in) ;

do

{

System.out.print("Enter a Number ------ ") ;

int x = sc.nextInt() ;

if(x > 9999 || x < 0)

System.out.println("Output ---- Invalid entry") ;

else

System.out.println("Output ---- " + t.towords(x)) ;

System.out.print("Want to Continue?(y/n)-------") ;

}

while(sc.next().charAt(0) != 'n') ;

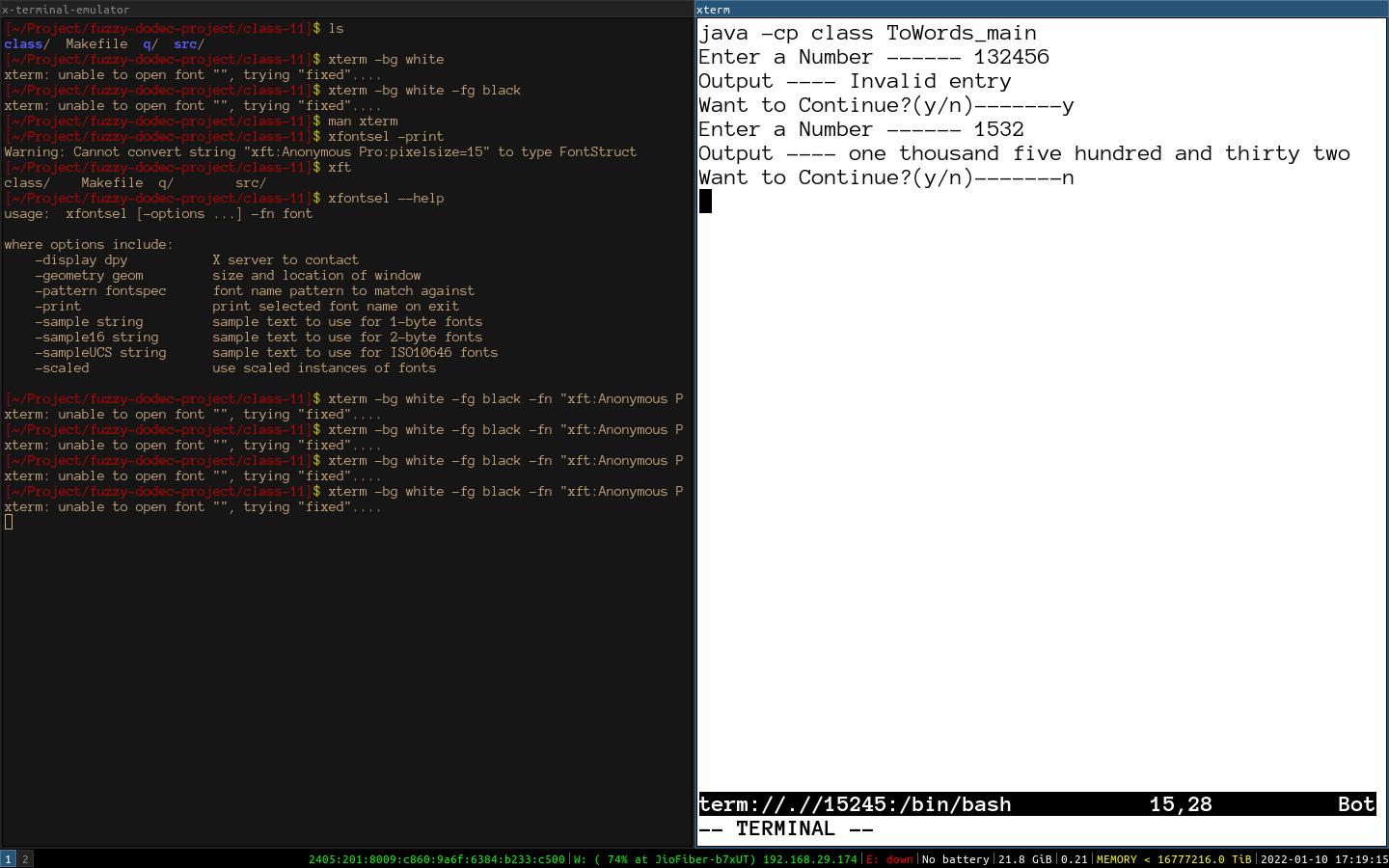
}

}

**Variable Listing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Function** | **Data Type** | **Scope** |
| ones | Ones look up table | String[] | ToWords |
| teen | Teens special case lookup table | String[] | ToWords |
| tens | Tens look up table | String[] | ToWords |
| out | Out is the strigified word | String | towords |
| d | Thousands face value | int | towords |
| h | Hundreds face value | int | towords |
| t | Tens facevalue | int | towords |
| o | Ones face value | int | towords |

**Output:**

**Assignment 19**

Write a Java program to input a sentence from the user in lowercase and removes the first and the last characters of every word in it.

Sample Input :i love java for school.

Sample Ouptut :ovav o choo

Some of the data members and member functions are given below:

Class name : Remove

Data members/instance variables:

sent : stores the sentence

rem : to store the new sentence

size : stores the length of the sentence

Member functions:

Remove() : default constructor

void readsentence() : to accept the sentence

void remfirstlast() : extract each word and remove the first and the last alphabet of the word and form a new sentence 'rem' using the changed words.

void display() : display the original sentence along with the new changed sentence.

Specify the class Remove giving details of the constructor Remove (), void readsentence(), void remfirstlast() and void display(). Define the main() function to create an object and call the function accordingly to enable the task.

**Algorithm:**

Class Remove:

Method readsentence:

Step 1: Create input handler

Step 2: accept a line as input

Step 3: set the size as the length of the line

Method remfirstlast:

Step 1: Extract the array of words from the string

Step 2: get word to iterate over words

Step 3: remove the first and the last chars of word string

Method display:

Step 1: print sentence

Step 2: print newline

Step 3: print rem

Class Remove\_main:

Method main:

Step 1: create object

Step 2: call readsentence

Step 3: call remfirstlast

Step 4: call displays

**Source Code:**

import java.util.Scanner ;

public class Remove

{

private String sent ; // stores the sentence

private String rem ; // to store the new sentence

private int size ; // stores the length of the sentence

Remove()

{

this.sent = "" ;

this.rem = "" ;

this.size = 0 ;

}

void readsentence()

{

Scanner sc = new Scanner(System.in) ;

this.sent = sc.nextLine() ;

this.size = this.sent.length() ;

}

void remfirstlast()

{

String words[] = this.sent.split(" ") ;

for(String word : words)

{

// should get something done OwO :)

if(word.length() < 2) continue ;

this.rem += word.substring(1, word.length()-1) + " ";

}

}

void display()

{

System.out.println(this.sent+"\n"+this.rem) ;

}

}

public class Remove\_main

{

public static void main(String args[])

{

Remove rem = new Remove() ;

rem.readsentence() ;

rem.remfirstlast() ;

rem.display() ;

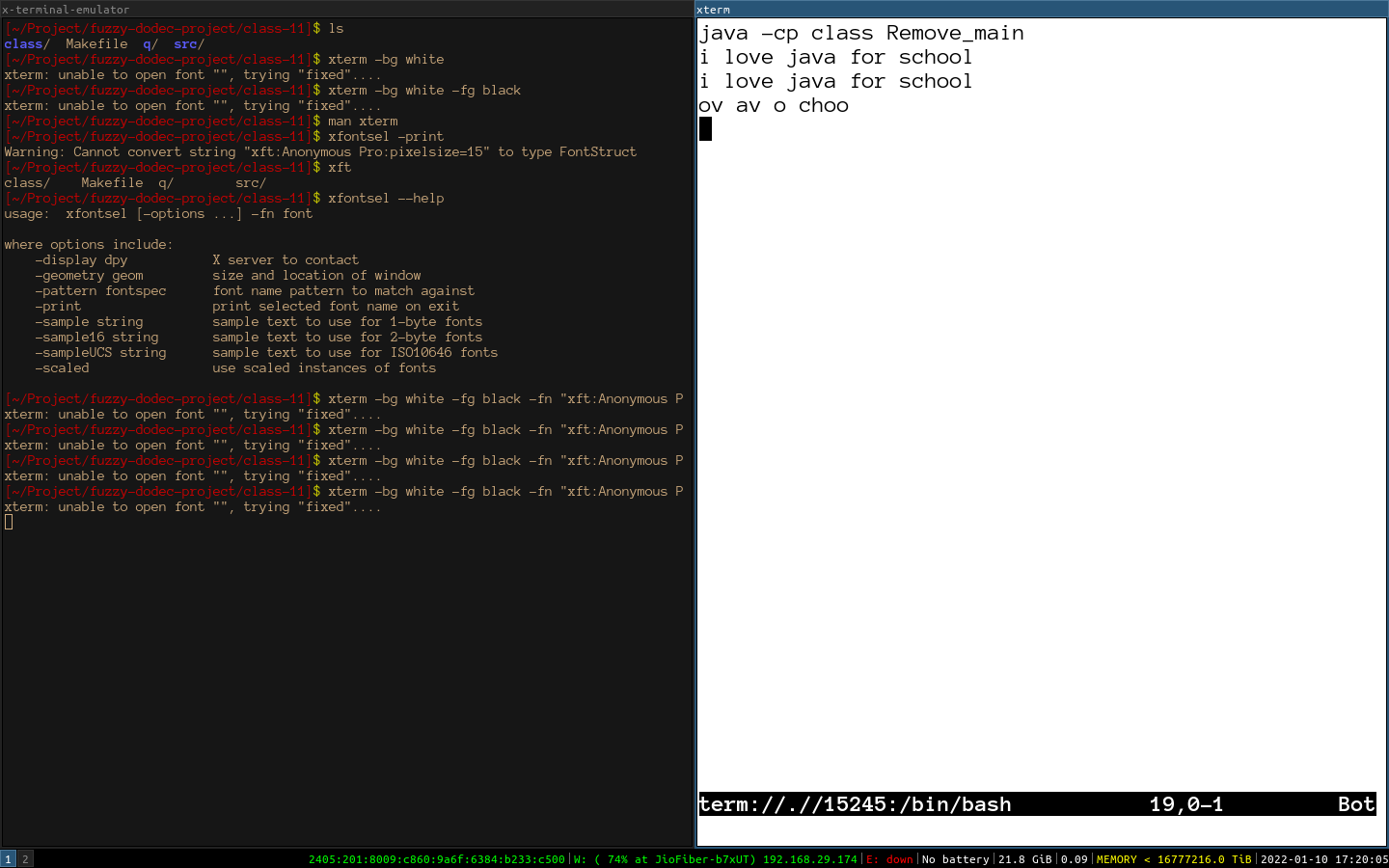
}

}

**Variable Listing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Function** | **Data Type** | **Scope** |
| sent | Stores the sentence | String | Remove |
| rem | To store the new sentence | String | Remove |
| size | Stores the length of the sentece | int | Remove |
| sc | Input handler | Scanner | readsentence |
| words | Array of words fromed from sent | String[] | remfirstlast |
| word | Iterator over words variable | String | readsentence |

**Output:**



**Assignment 20**

A sentence is terminated by either “.” , “!” or “?” followed by space. Input a piece of text consisting of sentences. Assume that there will be a maximum of 10 sentences in block letters. Write a program to: a) Obtain the length of the sentence (measured in words) and the frequency of vowels in each sentence. b) Generate the output as shown below using the given data.

**Algorithm:**

Class Sentence:

Method Sentece:

Step 1: split the string into sentences using regular expression [?!.] and the split method

Step 2: set the size of vow and words to be the same as the sentences

Method fill:

Step 1: loop through all the sentences

Step 2: set the vowels to be number of vowels in the sentences

Step 3: set the words to be the number words in the sentences

Method display:

Step1 : display the first part of statistic like the sample

Step2: display the second part of the statictic like given in sample

Class Sentence\_main:  
Method main:

Step 1: create input handler

Step 2: take input and create object

Step 3: call the fill and display methods

**Variable Listing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Function** | **Data Type** | **Scope** |
| sets | The sentences to be function upon | String[] | Sentence |
| vows | Vows present in sentences | int[] | Sentence |
| words | Words present in sentence | int[] | Sentence |
| i | Iterative variable | int | fill |
| j | Iterative variable | int | fill |

**Output:**

