

**BHARATI VIDYAPEETH’S**

**INSTITUTE OF COMPUTER APPLICATIONS & MANAGEMENT**

(Affiliated to Guru Gobind Singh Indraprastha University, Approved by AICTE, New Delhi)

**Object-Oriented Programming and Java**

**(MCA-167)**

**Practical File**

**Submitted To: Submitted By:**

Dr. Ritika Wason Garima Dixit

(Associate Professor) MCA 1st Sem, Sec 1/2

**I NDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Problem Description** | **Date of Execution** | **Sign.** |
| P1 | Explore the basic java program development scenario in Notepad++ and cmd by creating a program TenHelloWorlds.java. Your program should print HelloWorld ten times? (Explore the loop control structures) | 18-09-2023 |  |
| P2 | Explore the basic decision control structures to find the number of days in a month of a specific year? (Assess the solution using both decision control structures to assess which is more optimal) | 18-09-2023 |  |
| P3 | Write a program UserArgument.java that accepts three names from the user at command line and prints out a proper sentence with the names printed in reverse order? | 21-09-2023 |  |
| P4 | Develop a Number Reciprocator java application to computes the sum of the reciprocals with using a single variable in the format:  1/1 + 1/2 + 1/3 + ... + 1/10 | 21-09-2023 |  |
| P5 | Demonstrate type conversion in a simple java program by casting and checking output in the following cases:-  a) Conversion of int to byte  b) Conversion of double to int  c) Conversion of double to byte  d) Conversion of int to char  e) Conversion of float to short | 25-09-2023 |  |
| P6 | Construct a character counter that inputs a piece of text that is analyzed character by character to determine the vowels, spaces and letters used. Fill in the code that computes the number of spaces, vowels, and consonants.  public class StringCharacters {  public static void main(String[] args) {  String text = "To be or not to be, that is the question;"  +"Whether this nobler in the mind to suffer"  +" the slings and arrows of outrageous fortune,"  +" or to take arms against a sea of troubles,"  +" and by opposing end them?";  int spaces = 0, vowels = 0, letters = 0;  //YOUR CODE HERE  System.out.println("The text contained vowels: " + vowels + "\n" + consonants " + (letters - vowels) + "\n"+ spaces: " + spaces);  }} | 25-09-2023 |  |
| P7 | Create an array to hold certain integer elements entered by the user. Search for a given element desired by the user in this array using Linear Search? | 28-09-23 |  |
| P8 | Create a class named DuplicateFinder which initializes an array of at least 15 elements. Define appropriate methods to print its elements and calculate duplicate elements if any. It should detail the number of duplicates along with their frequency of occurrence. | 28-09-23 |  |
| P9 | A group of BVICAM friends decide to run the Airtel Delhi Half Marathon. Their names and times (in minutes) are below:  Name      Time (minutes)   Elena          341   Thomas      273  Hamilton    278   Suzie        329  Phil          445  Matt      402  Alex      388  Emma   275  John   243   James   334   Jane   412  Find the fastest runner. Print the name and his/her time (in minutes). Optional: Find the second fastest runner. Print the name and  his/her time (in minutes). | 28-09-23 |  |
| P10 | Define a class named VowelFilter which contains a static method named filterVowel(). This method receives a character array as argument and returns another array which contains only the nonvowel characters of the argument array | 28-09-23 |  |
| P11 | A word that reads the same backward as forward is called a palindrome, e.g., "mom", "dad", "racecar", "madam", and "Radar" (case-insensitive). Create a class called TestPalindromicWord, that prompts user for a word and prints ""xxx" is|is not a palindrome". | 28-09-23 |  |
| P12 | Demonstrate the working of a Static Inner Class through a class Electronics and within it create Static Inner Class Television that has a method cost() which displays cost of television object passed in constructor of Television class. Demonstrate invoking inner class method with outer object when the method cost() is once a :-  a) Instance(Non static) method  b) Static method | 5-10-23 |  |
| P13 | Simulate a simple banking application. Provide for classes BankAccount. Account will be of two type- Savings and Current. There should be methods to open an account, close an account and perform withdraw, deposit and transfer operations on an account as abstract methods in Account and properly overridden with definition in Account Types. Test classes should instantiate Account Type Classes and provide a menu driven option for operations. | 5-10-23 |  |
| P14 | Use ragged array to provide the output given below (Take row count from user).   1  12  123  1234  12345   123456   1234567 | 5-10-23 |  |
| P15 | Design a class called DecipherCaesarCode to decipher the Caesar's code. The program shall prompts user for a ciphertext string consisting of mix-case letters only; compute the plaintext; and print the plaintext in uppercase. Design the solution with appropriate methods? | 5-10-23 |  |
| P16 | An ExceptionPOC class is requesting a number between 1 and 10. Run the program again and enter 5.5. Although this number is between 1 and 10, the program will abort. Examine the error message. You should see the word Exception, the method where the exception occurred (main), the class name of the exception (InputMismatchException), as well as the call stack listing the method calls. Add a try/catch block to catch and handle the InputMismatchException exception. Identify the statements that cause the error as well as the portions of the program that depend upon these statements. Enclose these statements within the try block. Follow the try block with the catch block given below. Note, the InputMismatchException class is defined in java.util and must be imported. Also, when the Scanner throws an InputMismatchException, the input token will remain in the buffer so that it can be examined by the program. Complete code by implementing the same using:- a) Throws method declaration. b) Throw keyword. | 5-10-23 |  |
| P17 | A PalindromesViaThreads class that prints the prints 20 palindrome numbers(121,131,232) between 100 and 1000 using Threads implemented via:- a) Extending Thread Class  b) Implementing Runnable Interface | 5-10-23 |  |
| P18 | Extend your Banking Application exercise (Ques P13) to include the concept of Synchronization in a multithreaded environment using the approach of block synchronization. Methods to be made thread safe are:-  a) withdrawl()  b) deposit() |  |  |
| P19 | A result generator application maintains objects of class Result in a TreeSet. The ordering of objects is to be based on Semester and then roll no. The class has following members: a) Rollno: String b) Semester: String c) Array for marks in five subjects The program should produce the result per semester of roll-number sorted students in the following format: RollNo: ------ SEM MARKS1 MARKS2 MARKS3 MARKS4 MARKS5 Sum Total \_\_\_\_\_\_ | 5-10-23 |  |
| P20 | An Employee object, which has fields like int id, String name, int salary, int age and Date field to represent date of joining. Our requirement is to sort a List of employee based upon their name, age, salary and date of joining. | 5-10-23 |  |
| P21 | A connection-oriented client/server application using TCP/IP protocol where the client has the following responsibilities:  a) Server: Creates an Employee Class having fields- employeeName, employeeID and department. Server holds an array of employee objects.  b) Client: Accept the employeeID of an employee as an integer from the user.   c) Server: Searches for corresponding employee object, in the array and write its details to the client stream.  d) Client: Display the received object’s information | 5-10-23 |  |
| P22 | Create a connection-less Client/Server application using UDP protocol that sends system date and time in the format requested by the client. a) Client: Reads a string representing the required format from the enduser. b) Server: Returns the system date and time in the requested format or a default format if received format is not understandable. c) Client: Display the returned contents | 5-10-23 |  |

P1.Explore the basic java program development scenario in Notepad++ and cmd by creating a program TenHelloWorlds.java. Your program should print HelloWorld ten times? (Explore the loop control structures)

SOLUTION:

FOR LOOP

class TenHelloWorlds

{

public static void main (String args[])

{

//using for loop

for (int i = 0; i < 10; i++)

{

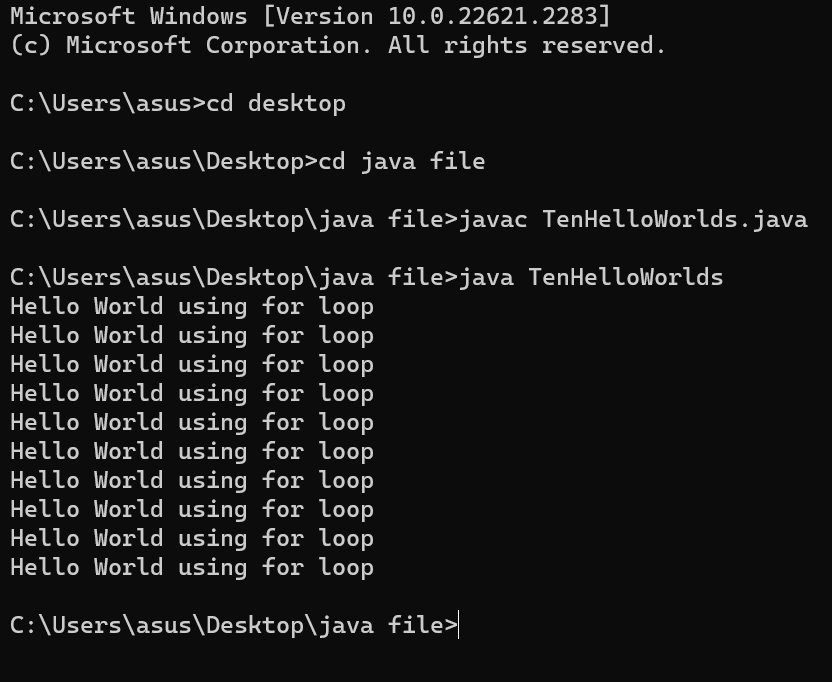
System.out.println ("Hello World using for loop");

}

}

}

CODE:



WHILE LOOP

class whileloop

{

public static void main (String[]args)

{

//using while loop

int i = 0;

while (i < 10)

{

System.out.println ("Hello World using while loop");

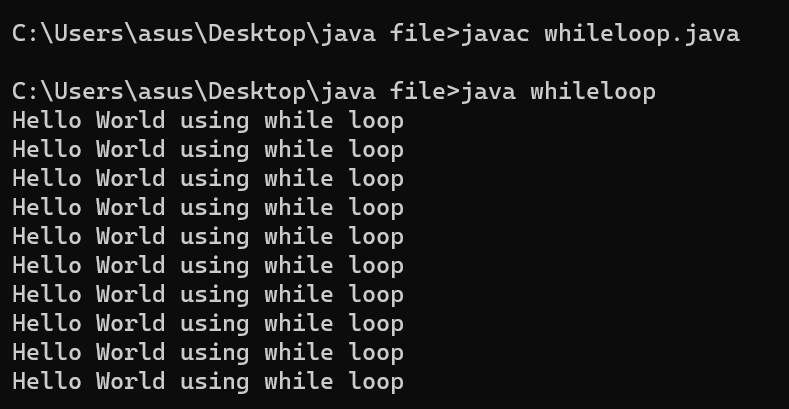
i++;

}

}

}

CODE:



DO WHILE LOOP:

class dowhileloop

{

public static void main (String[]args)

{

//using do while loop

int i = 0;

do

{

System.out.println ("Hello World using do while loop");

i++;

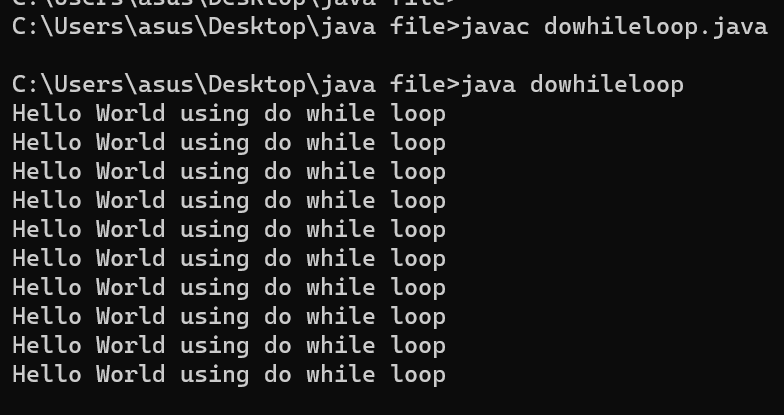
}

while (i < 10);

}

}

CODE:



P2.

Explore the basic decision control structures to find the number of days in a month of a specific year? (Assess the solution using both decision control structures to assess which is more optimal).

SOLUTION:

class month

{

public static void main(String[] args) {

int month = 12;

int year = 2024;

int numDays = 0;

switch(month)

{

case 1: case 3: case 5: case 7: case 8: case 10: case 12:

numDays=31;

break;

case 4: case 6: case 9: case 11:

numDays=30;

break;

case 2:

if((year%4==0 && year%100!=0) || (year%400==0))

numDays=29;

else

numDays=28;

break;

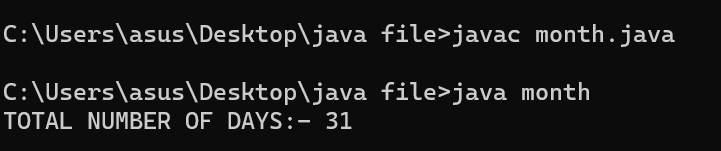
}

System.out.println("TOTAL NUMBER OF DAYS:- " +numDays);

}

}

CODE:



P3.

Write a program UserArgument.java that accepts three names from the user at command line and prints out a proper sentence with the names printed in reverse order?

SOLUTION:

import java.util.Scanner;

class names {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

String namel = in.next();

String name2 = in.next();

String name3 = in.next();

System.out.println("hello " + namel);

System.out.println("hello " + name2);

System.out.println("hello " + name3);

int x\_len = namel.length();

String x = "";

for (int i = x\_len - 1; i >= 0; i--) {

x += namel.charAt(i);

}

int y\_len = name2.length();

String y = "";

for (int i = y\_len - 1; i >= 0; i--) {

y += name2.charAt(i);

}

int z\_len = name3.length();

String z = "";

for (int i = z\_len - 1; i >= 0; i--) {

z += name3.charAt(i);

}

System.out.println("Reversed namel: " + x);

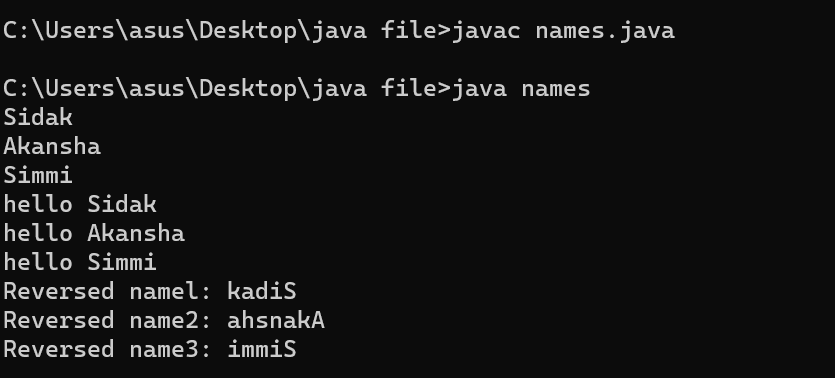
System.out.println("Reversed name2: " + y);

System.out.println("Reversed name3: " + z);

}

}

CODE:



P4.

Develop a Number Reciprocator java application to computes the sum of the reciprocals with using a single variable in the format:

1/1 + 1/2 + 1/3 + ... + 1/10

SOLUTION:

class reciprocal {

public static void main(String[] args) {

double sum = 0.0;

for (int i = 1; i <= 10; i++) {

sum += 1.0 / i;

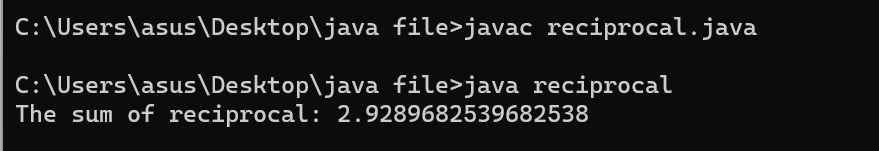
}

System.out.println("The sum of reciprocal: " + sum);

}

}

CODE:



P5:Demonstrate type conversion in a simple java program by casting and checking output in the following cases:-

a) Conversion of int to byte

b) Conversion of double to int

c) Conversion of double to byte

d) Conversion of int to char

e) Conversion of float to short

SOLUTION:

public class size {

public static void main(String[] args) {

// Conversion of int to byte

int intValue = 100;

byte byteValue = (byte) intValue;

System.out.println("a) int to byte: " + byteValue);

// Conversion of double to int

double doubleValue = 123.456;

int intValue2 = (int) doubleValue;

System.out.println("b) double to int: " + intValue2);

// Conversion of double to byte

double doubleValue2 = 200.789;

byte byteValue2 = (byte) doubleValue2;

System.out.println("c) double to byte: " + byteValue2);

// Conversion of int to char

int intValue3 = 65;

char charValue = (char) intValue3;

System.out.println("d) int to char: " + charValue);

// Conversion of float to short

float floatValue = 123.45f;

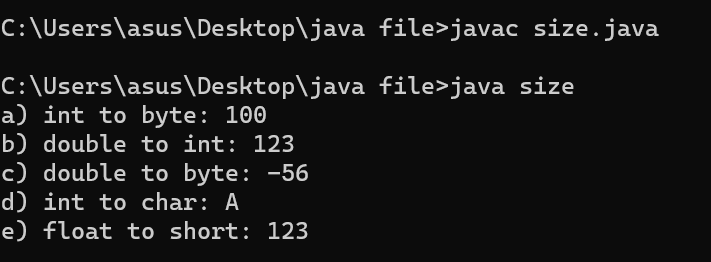
short shortValue = (short) floatValue;

System.out.println("e) float to short: " + shortValue);

}

}

CODE:



P6:Construct a character counter that inputs a piece of text that is analyzed character by character to determine the vowels, spaces and letters used. Fill in the code that computes the number of spaces, vowels, and consonants.

public class StringCharacters {

public static void main(String[] args) {

String text = "To be or not to be, that is the question;"

+"Whether this nobler in the mind to suffer"

+" the slings and arrows of outrageous fortune,"

+" or to take arms against a sea of troubles,"

+" and by opposing end them?";

int spaces = 0, vowels = 0, letters = 0;

//YOUR CODE HERE

System.out.println("The text contained vowels: " + vowels + "\n" + consonants " + (letters - vowels) + "\n"+ spaces: " + spaces);

}}

SOLUTION:

public class alphabets {

public static void main(String[] args) {

String text = "To be or not to be, that is the question;"

+ "Whether this nobler in the mind to suffer"

+ " the slings and arrows of outrageous fortune,"

+ " or to take arms against a sea of troubles,"

+ " and by opposing end them?";

int spaces = 0, vowels = 0, consonants = 0;

// Convert the input text to lowercase for case-insensitive matching

text = text.toLowerCase();

// Iterate through each character in the text

for (char ch : text.toCharArray()) {

// Check if the character is a space

if (ch == ' ') {

spaces++;

}

// Check if the character is a vowel

else if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {

vowels++;

}

// Check if the character is a letter (consonant)

else if (ch >= 'a' && ch <= 'z') {

consonants++;

}

}

System.out.println("The text contained vowels: " + vowels);

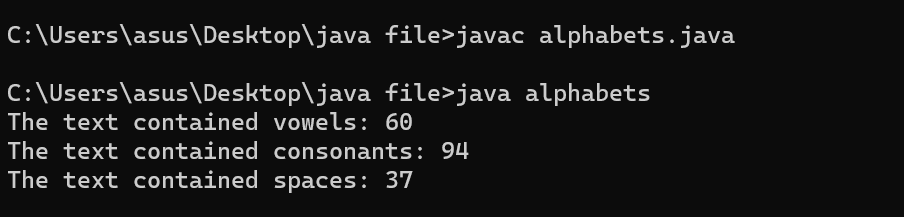
System.out.println("The text contained consonants: " + consonants);

System.out.println("The text contained spaces: " + spaces);

}

}

CODE:



P7: Create an array to hold certain integer elements entered by the user. Search for a given element desired by the user in this array using Linear Search?

CODE:

import java.util.\*;

public class P7 {

public static void main(String[] args) {

int size = 0, search = 0, i = 0;

boolean found = false;

Scanner sc = new Scanner(System.in);

System.out.print("Enter the number of elements you want to enter in the array: ");

size = sc.nextInt();

int[] arr = new int[size];

for (i = 0; i < size; i++) {

System.out.println("\nEnter the number: ");

arr[i] = sc.nextInt();

sc.nextLine();

}

System.out.print("\nEnter the number you want to search in the array: ");

search = sc.nextInt();

for (i = 0; i < size; i++) {

if (search == arr[i]) {

System.out.print("\n" + search + " is present in the array at index " + i);

found = true;

}

}

if (!found) {

System.out.print(search + " is not present in the array.");

}

}

}

OUTPUT:

A computer screen shot of a black screen

Description automatically generated

P8: Create a class named DuplicateFinder which initializes an array of at least 15 elements. Define appropriate methods to print its elements and calculate duplicate elements if any. It should detail the number of duplicates along with their frequency of occurrence.

CODE:

import java.util.\*;  
  
public class P8 {  
 static boolean inArray(int num, int[] arr) {  
 if (arr.length == 0) {  
 return false;  
 }  
  
 for (int i = 0; i < arr.length; i++) {  
 if (num == arr[i]) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 static void printArray(int[] arr) {  
 Scanner sc = new Scanner(System.*in*);  
 for (int i = 0; i < arr.length; i++) {  
 System.*out*.print(arr[i] + " ");  
 }  
 }  
  
 static void DuplicateFinder(int[] arr) {  
 int count = 0, k = 0;  
 int[] written = new int[15];  
 for (int i = 0; i < (arr.length); i++) {  
 for (int j = 0; j < arr.length; j++) {  
 if (arr[i] == arr[j]) {  
 count++;  
 }  
 }  
 if (!*inArray*(arr[i], written)) {  
 System.*out*.println("\n" + arr[i] + " appears in the array " + count + " times.");  
 written[k] = arr[i];  
 k++;  
 }  
 count = 0;  
 }  
 }  
  
 public static void main(String args[]) {  
 int[] ar = {1, 2, 3, 4, 5, 7, 8, 9, 1, 2, 3, 3, 3, 4, 4};  
  
*printArray*(ar);  
*DuplicateFinder*(ar);  
  
 }  
}

OUTPUT:

A screen shot of a computer program

Description automatically generated

P9: A group of BVICAM friends decide to run the Airtel Delhi Half Marathon. Their names and times (in minutes) are below:

Name      Time (minutes)

 Elena          341

 Thomas      273

Hamilton    278

 Suzie        329

Phil          445

Matt      402

Alex      388

Emma   275

John   243

 James   334

 Jane   412

Find the fastest runner. Print the name and his/her time (in minutes). Optional: Find the second fastest runner. Print the name and  his/her time (in minutes).

CODE: public class bvicam {

public static void main(String args[]) {

String[] name = {"Elena", "Thomas", "Hamilton", "Suzie", "Phil", "Matt", "Alex", "Emma", "John", "James", "Jane"};

int[] time = {341, 273, 278, 329, 445, 402, 388, 275, 243, 334, 412};

int min\_idx = 0, min = 1000, second = 1000, second\_idx = 0;

if (time[0] < time[1]) {

min = time[0];

second = time[1];

second\_idx = 1;

}

else{

min = time[1];

second = time[0];

second\_idx = 0;

}

for (int i = 2; i < time.length; i++) {

if (time[i] < min) {

second = min;

second\_idx = min\_idx;

min = time[i];

min\_idx = i;

} else if (time[i] < second) {

second = time[i];

second\_idx = i;

}

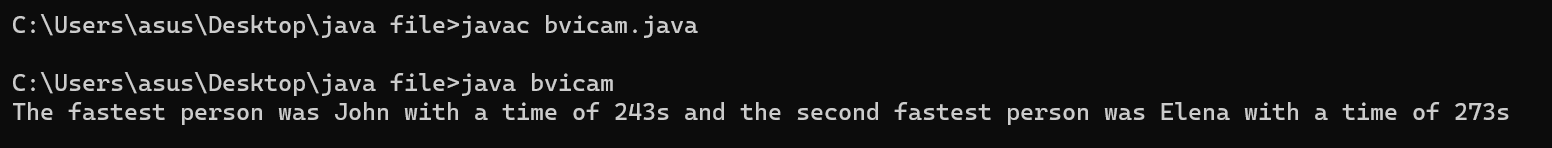
}

System.out.println("The fastest person was " + name[min\_idx] + " with a time of " + min + "s and the second fastest person was " + name[second\_idx] + " with a time of " + second + "s");

}

}

OUTPUT:



P10:

Define a class named VowelFilter which contains a static method named filterVowel(). This method receives a character array as argument and returns another array which contains only the nonvowel characters of the argument array.

CODE:

public class vowel {

static char[] filterVowel(char[] input) {

char[] word = new char[input.length];

int i = 0, j = 0;

for (char c : input) {

if (!((c == 'a') || (c == 'e') || (c == 'i') || (c == 'o') || (c == 'u'))) {

word[j] = input[i];

j++;

}

i++;

}

return word;

}

public static void main(String args[]) {

char[] word = {'s', 'i', 'd', 'a', 'k', 'a', 'h', 'u', 'j', 'a'};

char[] vowel\_removed\_word = new char[word.length];

vowel\_removed\_word = filterVowel(word);

for (char c : vowel\_removed\_word) {

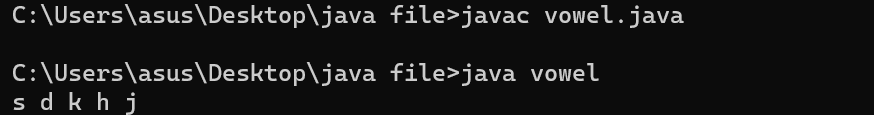
System.out.print(c + " "); // how to stop printing null characters?

}

}

}

OUTPUT:



P11:

A word that reads the same backward as forward is called a palindrome, e.g., "mom", "dad", "racecar", "madam", and "Radar" (case-insensitive). Create a class called TestPalindromicWord, that prompts user for a word and prints ""xxx" is|is not a palindrome".

CODE:

import java.util.\*;  
  
public class P11 {  
 static boolean checkPallindrome(String word) {  
 int start = 0, end = word.length() - 1, mid = 0;  
  
 mid = (end % 2 == 0) ? (end / 2) : (end / 2) + 1;  
  
 for (int i = 0; i <= mid; i++) {  
 if (Character.*toLowerCase*(word.charAt(start)) != Character.*toLowerCase*(word.charAt(end))) {  
 return false;  
 }  
 }  
  
 return true;  
 }  
  
 public static void main(String args[]) {  
 Scanner sc = new Scanner(System.*in*);  
 String word = "";  
 boolean result = false;  
  
 System.*out*.print("Enter a word: ");  
  
 word = sc.nextLine();  
  
 result = *checkPallindrome*(word);  
  
 if (result) {  
 System.*out*.println("The word " + word + " is a pallindrome!");  
 } else {  
 System.*out*.println("The word " + word + " is not a pallindrome!");  
 }  
 }  
}

OUTPUT:

A screen shot of a computer program

Description automatically generated

P12: Demonstrate the working of a Static Inner Class through a class Electronics and within it create Static Inner Class Television that has a method cost() which displays cost of television object passed in constructor of Television class. Demonstrate invoking inner class method with outer object when the method cost() is once a :-

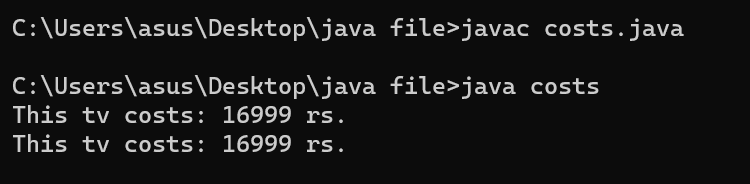
a) Instance(Non static) method

b) Static method

CODE:

public class costs {  
 public static void main(String args[]) {  
 //Electronics e1 = new Electronics();  
 //e1.Television.cost();  
 Electronics e1 = new Electronics();  
 Electronics.Television t1 = new Electronics.Television(16999);  
  
 e1.instanceCost(t1);  
 Electronics.*staticCost*(t1);  
 }  
}  
  
class Electronics {  
 static class Television {  
 private int cost = 0;  
  
 public Television(int cost) {  
 this.cost = cost;  
 }  
  
 public void cost() {  
 System.*out*.println("This tv costs: " + this.cost + " rs.");  
 }  
 }  
  
 public void instanceCost(Television t) {  
 t.cost();  
 }  
  
 public static void staticCost(Television t) {  
 t.cost();  
 }  
}

OUTPUT:



**P13:** Simulate a simple banking application. Provide for classes BankAccount. Account will be of two type- Savings and Current. There should be methods to open an account, close an account and perform withdraw, deposit and transfer operations on an account as abstract methods in Account and properly overridden with definition in Account Types. Test classes should instantiate Account Type Classes and provide a menu driven option for operations.

CODE:

import java.util.\*;

public class banking {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String name = "", aadhar = "", mobile = "", choice = "", type = "";

int amount = 0;

boolean opened = false, in = true;

Account a1 = new Account();

System.out.println("Welcome to Java Banking System!");

//test account

Account test = new Account();

test.openAccount("Garima Dixit", "2223243421", "987654314", "Current", "456");

test.deposit(15000);

while (true) {

if (!opened) {

System.out.println("\n\nAccounts currently open in our bank account: " + Account.accounts\_open);

System.out.println("Select a choice!");

System.out.println("1. Open a bank account");

System.out.println("If you wish to exit, press 'Y'");

choice = sc.nextLine();

switch (choice) {

case "1" -> {

// asking for basic details(couldn't make it a function as java doesn't allow multiple return variables)

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

name = sc.nextLine();

System.out.print("\nEnter your Aadhar number: ");

aadhar = sc.nextLine();

System.out.print("\nEnter your Mobile number: ");

mobile = sc.nextLine();

while (in) {

System.out.print("\nEnter the type of account you want to open(Savings(S)/Current(C)):");

type = sc.nextLine();

switch (type) {

case "S", "s" -> {

a1.openAccount(name, aadhar, mobile, "Savings");

opened = true;

in = false;

}

case "C", "c" -> {

a1.openAccount(name, aadhar, mobile, "Current");

opened = true;

in = false;

}

default -> System.out.println("Enter a valid choice!");

}

}

}

case "Y" -> System.exit(0);

default -> System.out.println("Enter a valid choice");

}

} else if (opened) {

System.out.println("\n\nSelect a choice!");

System.out.println("1. Withdraw money from your bank account");

System.out.println("2. Deposit money to your bank account");

System.out.println("3. Transfer money to another bank account");

System.out.println("4. Show account balance");

System.out.println("5. Close a bank account");

System.out.println("If you wish to exit, press 'Y'");

choice = sc.nextLine();

switch (choice) {

case "1" -> {

System.out.println("Enter the amount you want to withdraw:");

amount = sc.nextInt();

sc.nextLine();

a1.withdraw(amount);

}

case "2" -> {

System.out.println("Enter the amount you want to deposit:");

amount = sc.nextInt();

sc.nextLine();

a1.deposit(amount);

}

case "3" -> {

System.out.println("Enter the amount you want to transfer:");

amount = sc.nextInt();

sc.nextLine();

a1.transfer(test, amount);

}

case "4" -> System.out.println("Balance: " + a1.balance + " Rs.");

case "5" -> {

boolean status = false;

a1.withdraw(a1.balance);

status = a1.closeAccount();

if (status) {

System.gc();

opened = false;

in = true;

}

}

case "Y", "y" -> System.exit(0);

default -> System.out.println("Enter a valid choice");

}

}

}

}

}

abstract class BankAccount {

abstract void openAccount(String name, String aadhar, String mobile, String type, String password);

abstract void openAccount(String name, String aadhar, String mobile, String type);

abstract boolean closeAccount();

abstract void withdraw(int amount);

abstract void deposit(int amount);

abstract void transfer(Account a1, int amount);

}

class Account extends BankAccount {

Scanner sc = new Scanner(System.in);

String name = "", aadhar = "", mobile = "", type = "";

String password = "";

int balance = 0;

static int accounts\_open = 0;

void printBalance() {

System.out.println("\nThe current balance of " + name + "'s account is " + balance + " Rs.");

}

void printDetails() {

System.out.println("Name: " + name);

System.out.println("Aadhar Number: " + aadhar);

System.out.println("Mobile Number: " + mobile);

System.out.println("Account Type: " + type);

printBalance();

}

void openAccount(String name, String aadhar, String mobile, String type) {

this.name = name;

this.aadhar = aadhar;

this.mobile = mobile;

this.type = type;

System.out.print("Set your account password: ");

password = sc.nextLine();

accounts\_open++;

printDetails();

}

void openAccount(String name, String aadhar, String mobile, String type, String password) {

this.name = name;

this.aadhar = aadhar;

this.mobile = mobile;

this.type = type;

this.password = password;

accounts\_open++;

printDetails();

}

void deposit(int amount) {

if (amount <= 0) {

System.out.println("Amount to be added must be > 0!");

} else {

balance += amount;

System.out.println("\nDeposited " + amount + " Rs. in your account.");

}

printBalance();

}

void withdraw(int amount) {

if (amount > balance) {

System.out.println("Not sufficient balance in your account to withdraw " + amount + " Rs.");

System.out.println("Balance: " + balance + " Rs.");

} else if (amount <= 0) {

System.out.println("Withdrawal amount cannot be less than 0");

} else {

balance -= amount;

System.out.println("\nWithdrew " + amount + " Rs. from your account.");

printBalance();

}

}

void transfer(Account a1, int amount) {

if (amount > this.balance) {

System.out.println("Not sufficient balance in your account to transfer " + amount + " Rs.");

System.out.println("Balance: " + balance + " Rs.");

} else if (amount <= 0) {

System.out.println("Transfer amount cannot be less than 0 Rs.");

} else {

this.balance -= amount;

a1.balance += amount;

System.out.println("Successfully transfered " + amount + "Rs. from " + this.name + " to " + a1.name);

printBalance();

a1.printBalance();

}

}

boolean closeAccount() {

String pass;

System.out.println("Account Details");

printDetails();

System.out.println("\nTo close your account, please enter your password:");

pass = sc.nextLine();

if (pass.equals(password)) {

System.out.println("Your account has been closed.");

accounts\_open--;

return true;

} else {

System.out.println("Invalid password!");

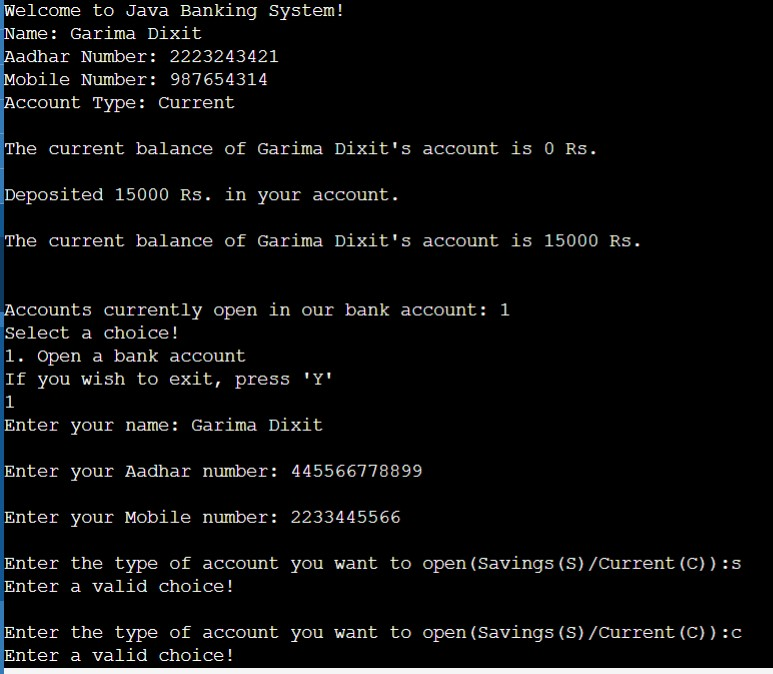
return false;

}

}

}

**OUTPUT:**



**P14:** Use ragged array to provide the output given below (Take row count from user).

 1

12

123

1234

12345

 123456

 1234567

CODE:

import java.util.\*;

public class ragged {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int rows;

System.out.println("Enter the number of rows:");

rows = sc.nextInt();

sc.nextLine();

int[][] arr = new int[rows][];

for (int i = 0; i < rows; i++) {

arr[i] = new int[i + 1];

for (int j = 0; j < i + 1; j++) {

arr[i][j] = j + 1;

}

}

for (int i = 0; i < rows; i++) {

for (int j = 0; j < i + 1; j++) {

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

}

System.out.println();

}

}

}

**OUTPUT:**

A computer screen with white text

Description automatically generated

P15:

Design a class called DecipherCaesarCode to decipher the Caesar's code. The program shall prompts user for a ciphertext string consisting of mix-case letters only; compute the plaintext; and print the plaintext in uppercase. Design the solution with appropriate methods?

CODE:

import java.util.\*;

public class Decipherceaserclass {

static String ceaserCipher(String plain\_text, boolean encode) {

Random r1 = new Random();

char ch;

int c;

String out = "";

plain\_text = plain\_text.toUpperCase();

int shift = 10;

boolean l1 = false, l2 = false;

for (int i = 0; i < plain\_text.length(); i++) {

ch = plain\_text.charAt(i);

if (Character.isAlphabetic(ch)) {

for (int j = 0; j < shift; j++) {

if (ch == 'Z' && !l1) {

ch = 'A';

l2 = true;

} else if (ch == 'A' && !l2) {

ch = 'Z';

l1 = true;

} else {

if (encode) {

ch++;

} else {

ch--;

}

}

}

}

out += ch;

}

return out;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String cipher = "", plain = "";

System.out.println("Enter the text: ");

plain = sc.nextLine();

cipher = ceaserCipher(plain, true);

System.out.println("Plain Text: " + plain);

System.out.println("Cipher Text: " + cipher);

plain = ceaserCipher(cipher, false);

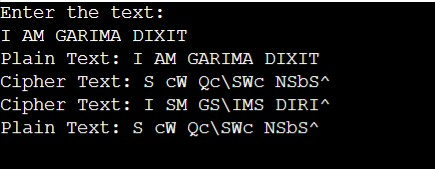
System.out.println("Cipher Text: " + plain);

System.out.println("Plain Text: " + cipher);

}

}

**OUTPUT:**



**P16**: An ExceptionPOC class is requesting a number between 1 and 10. Run the program again and enter 5.5. Although this number is between 1 and 10, the program will abort. Examine the error message. You should see the word Exception, the method where the exception occurred (main), the class name of the exception (InputMismatchException), as well as the call stack listing the method calls. Add a try/catch block to catch and handle the InputMismatchException exception. Identify the statements that cause the error as well as the portions of the program that depend upon these statements. Enclose these statements within the try block. Follow the try block with the catch block given below. Note, the InputMismatchException class is defined in java.util and must be imported. Also, when the Scanner throws an InputMismatchException, the input token will remain in the buffer so that it can be examined by the program. Complete code by implementing the same using:- a) Throws method declaration. b) Throw keyword.

CODE:

import java.util.\*;

public class exception {

// static void requestNumber() throws InputMismatchException{

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

//

// System.out.print("Enter a number between 1-10: ");

//

// try{

// int num = sc.nextInt();

// }

// catch (Exception e){

// System.out.println("Caught exception of type: " + e);

// }

//

// }

static void requestNumber() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a number between 1-10: ");

float num = sc.nextFloat();

try {

if (num > 10 || num < 1) {

throw new InputMismatchException("The number should be between 1-10!");

}

// if number is not an integer

else if (num % 1 != 0) {

throw new InputMismatchException("The number should not be a decimal number!");

}

}

catch (Exception e){

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

}

}

public static void main(String[] args) {

requestNumber();

}

}

OUTPUT:

A black screen with white text

Description automatically generated

**P17**: A PalindromesViaThreads class that prints the prints 20 palindrome numbers(121,131,232) between 100 and 1000 using Threads implemented via:- a) Extending Thread Class

b) Implementing Runnable Interface

CODE:

import java.util.concurrent.Executor;

public class PalindromesViaThreadsP17 implements Runnable {

static int count=0;

public static boolean checkPalindrome(int i){

int num = i, out = 0;

while (i > 0){

out = (out\*10) + i%10;

i = i/10;

}

return num == out;

}

public void run(){

for(int i=100; i<1001; i++) {

if (count >=20){

break;

}

if (checkPalindrome(i)) {

System.out.println(i + " Name: " + Thread.currentThread().getName());

count++;

}

}

}

public static void main(String[] args) {

PalindromesViaThreadsP17 t = new PalindromesViaThreadsP17();

Thread t1 = new Thread(t, "First Thread");

Thread t2 = new Thread(t, "Second Thread");

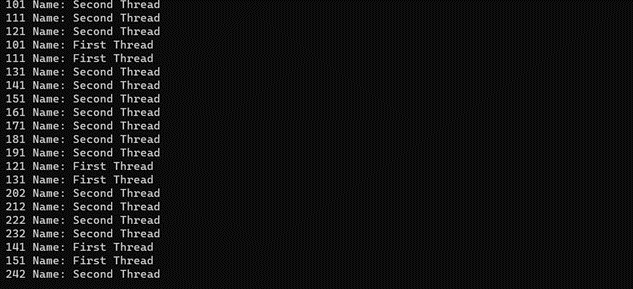
t1.start();

t2.start();

}

}

**OUTPUT:**



**P18:** Extend your Banking Application exercise (Ques P13) to include the concept of Synchronization in a multithreaded environment using the approach of block synchronization. Methods to be made thread safe are:-

a) withdrawl()

b) deposit()

P19: A result generator application maintains objects of class Result in a TreeSet. The ordering of objects is to be based on Semester and then roll no. The class has following members: a) Rollno: String b) Semester: String c) Array for marks in five subjects The program should produce the result per semester of roll-number sorted students in the following format: RollNo: ------ SEM MARKS1 MARKS2 MARKS3 MARKS4 MARKS5 Sum Total \_\_\_\_\_\_

CODE:

import java.util.NavigableSet;

import java.util.TreeSet;

public class ResultGenerator {

static class Result implements Comparable<Result>{

String rollno, semester;

int[] marks;

int total=0;

Integer tot;

public Result(String rollno, String semester, int[] marks) {

this.rollno = rollno;

this.semester = semester;

this.marks = marks;

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

this.total += marks[i];

}

this.tot = this.total;

}

@Override

public int compareTo(Result R) {

if (this.semester.equals(R.semester)){

if (this.rollno.equals(R.rollno)){

return R.tot.compareTo(this.tot);

}

return Integer.valueOf(this.rollno).compareTo(Integer.valueOf(R.rollno));

}

else {

return Integer.valueOf(this.semester).compareTo(Integer.valueOf(R.semester));

}

}

@Override

public String toString() {

return "\nSemester: " + this.semester + "\n" + "Roll No.: " + this.rollno + "\n" + this.marks[0] + " " +

this.marks[1] + " " + this.marks[2] + " " + this.marks[3] + " " + this.marks[4] + "\n" +

"Total Marks: " + this.total + "/500";

}

}

public static void main(String[] args) {

int[] m1 = {100, 80, 75, 55, 58};

int[] m2 = {60, 80, 65, 72, 99};

int[] m3 = {62, 90, 87, 75, 88};

int[] m4 = {78, 70, 65, 95, 81};

int[] m5 = {65, 71, 97, 83, 89};

Result r1 = new Result("1", "1", m1);

Result r2 = new Result("1", "1", m2);

Result r3 = new Result("3", "2", m3);

Result r4 = new Result("4", "3", m4);

Result r5 = new Result("5", "3", m5);

NavigableSet<Result> ts = new TreeSet<>();

ts.add(r1);

ts.add(r2);

ts.add(r3);

ts.add(r4);

ts.add(r5);

for (Result ele: ts){

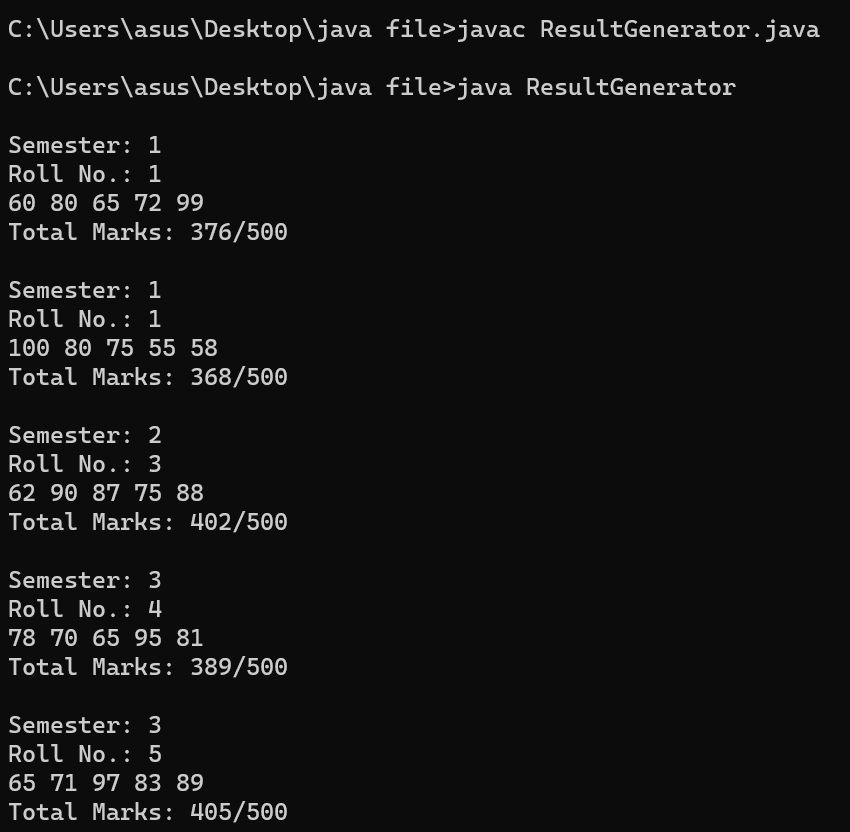
System.out.println(ele);

}

}

}

**OUTPUT:**



**P20**: An Employee object, which has fields like int id, String name, int salary, int age and Date field to represent date of joining. Our requirement is to sort a List of employee based upon their name, age, salary and date of joining.

CODE:

import java.util.\*;

public class EmployeeP20 implements Comparable<EmployeeP20>{

Integer id, salary, age;

String name;

Integer[] DOJ;

EmployeeP20(int id, String name, int age, int salary, Integer[] DOJ) {

this.id = id;

this.name = name;

this.age = age;

this.salary = salary;

this.DOJ = DOJ;

}

@Override

public String toString() {

return "\nID: " + this.id + "\nName: " + this.name + "\nAge: " + this.age + "\nSalary: " + this.salary +

"\nDate of Joining: " + this.DOJ[0] + "/" + this.DOJ[1] + "/" + this.DOJ[2];

}

@Override

public int compareTo(EmployeeP20 E) {

return this.id.compareTo(E.id);

}

// currently these comparators don't know what to do when the values are equal, try and find the proper way to deal

// with this and implement it and if we can change comparator afterwards without just creating a new tree set

static class NameSort implements Comparator<EmployeeP20> {

public int compare(EmployeeP20 a, EmployeeP20 b)

{

return a.name.compareTo(b.name);

}

}

static class AgeSort implements Comparator<EmployeeP20> {

public int compare(EmployeeP20 a, EmployeeP20 b)

{

return a.age.compareTo(b.age);

}

}

static class SalarySort implements Comparator<EmployeeP20> {

public int compare(EmployeeP20 a, EmployeeP20 b)

{

return a.salary.compareTo(b.salary);

}

}

static class DOJSort implements Comparator<EmployeeP20> {

public int compare(EmployeeP20 a, EmployeeP20 b)

{

if (a.DOJ[2].compareTo(b.DOJ[2]) == 0){

if (a.DOJ[1].compareTo(b.DOJ[1]) == 0){

if (a.DOJ[0].compareTo(b.DOJ[0]) == 0){

return a.id.compareTo(b.id);

}

else{

return a.DOJ[0].compareTo(b.DOJ[0]);

}

}

return a.DOJ[1].compareTo(b.DOJ[1]);

}

return a.DOJ[2].compareTo(b.DOJ[2]);

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int choice;

SortedSet<EmployeeP20> ss = null;

Integer[] d1 = {5, 9, 2023};

Integer[] d2 = {9, 9, 2020};

Integer[] d3 = {11, 12, 2015};

Integer[] d4 = {15, 6, 2019};

Integer[] d5 = {25, 6, 2019};

EmployeeP20 e1 = new EmployeeP20(1, "Sidak Ahuja", 22, 100000, d1);

EmployeeP20 e2 = new EmployeeP20(2, "Simran Arora", 28, 130000, d2);

EmployeeP20 e3 = new EmployeeP20(3, "Deepak Oberoi", 26, 180000, d3);

EmployeeP20 e4 = new EmployeeP20(4, "Kanan Sharma", 30, 230000, d4);

EmployeeP20 e5 = new EmployeeP20(5, "Aman Malhotra", 23, 99000, d5);

System.out.println("Enter Sorting criteria: ");

System.out.println("1. Name");

System.out.println("2. Age");

System.out.println("3. Salary");

System.out.println("4. Date of joining");

choice = sc.nextInt();

switch (choice) {

case 1 -> ss = new TreeSet<>(new NameSort());

case 2 -> ss = new TreeSet<>(new AgeSort());

case 3 -> ss = new TreeSet<>(new SalarySort());

case 4 -> ss = new TreeSet<>(new DOJSort());

default -> {

System.out.println("Enter a valid option!");

System.exit(0);

}

}

ss.add(e1);

ss.add(e2);

ss.add(e3);

ss.add(e4);

ss.add(e5);

for (EmployeeP20 e : ss) {

System.out.println(e);

}

}

}

**OUTPUT**:

A computer screen shot of a black screen

Description automatically generated

A screenshot of a computer

Description automatically generated

**P21**:

A connection-oriented client/server application using TCP/IP protocol where the client has the following responsibilities:

a) Server: Creates an Employee Class having fields- employeeName, employeeID and department. Server holds an array of employee objects.

b) Client: Accept the employeeID of an employee as an integer from the user.

 c) Server: Searches for corresponding employee object, in the array and write its details to the client stream.

d) Client: Display the received object’s information.

CODE:

SERVER:

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.net.ServerSocket;

import java.net.Socket;

public class P21Server {

static class Employee{

public static int empCount = 0;

String empName, empDepartment;

int empID;

Employee(String name, String dept, int id){

this.empName = name;

this.empDepartment = dept;

this.empID = id;

empCount++;

}

@Override

public String toString() {

return "\nEmployee Name: " + this.empName + "\nEmployee ID: " + this.empID +

"\nEmployee Department: " + this.empDepartment;

}

}

public static void main(String[] args){

Employee e1 = new Employee("Sidak Ahuja", "SDE", 101);

Employee e2 = new Employee("Pratishtha Arora", "ST", 102);

Employee e3 = new Employee("Vinny Malhotra", "Sales", 103);

Employee e4 = new Employee("Honey", "Morale", 104);

Employee e5 = new Employee("Arleen Kaur", "US", 105);

Employee[] empList = {e1, e2, e3, e4, e5};

String out\_message="";

try{

ServerSocket ss = new ServerSocket(8393);

Socket s = ss.accept();

DataInputStream dis = new DataInputStream(s.getInputStream());

DataOutputStream dos = new DataOutputStream(s.getOutputStream());

int id = Integer.parseInt(dis.readUTF());

System.out.println("ID: " + id);

for(int i=0; i<Employee.empCount; i++){

if (empList[i].empID == id){

out\_message = empList[i].toString();

break;

}

else{

out\_message = "No employee witn that ID exists in our database!";

}

}

dos.writeUTF(out\_message);

dos.flush();

dos.close();

ss.close();

}

catch(Exception e){

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

}

}

}

CLIENT:

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.net.Socket;

import java.util.InputMismatchException;

import java.util.Scanner;

public class P21Client {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int id=0;

try{

System.out.print("Enter the empID of employee whose details you want to get: ");

id = sc.nextInt();

}

catch (InputMismatchException e){

System.out.println("Please enter an integer!");

System.exit(0);

}

try{

Socket s = new Socket("localhost",8393);

DataOutputStream dout = new DataOutputStream(s.getOutputStream());

dout.writeUTF(Integer.toString(id));

dout.flush();

DataInputStream dis = new DataInputStream(s.getInputStream());

System.out.println(dis.readUTF());

s.close();

dout.close();

}

catch(Exception e){

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

}

}

}

**OUTPUT:**

A black background with white text

Description automatically generated

A screen shot of a computer program

Description automatically generated

**P22**: Create a connection-less Client/Server application using UDP protocol that sends system date and time in the format requested by the client. a) Client: Reads a string representing the required format from the enduser. b) Server: Returns the system date and time in the requested format or a default format if received format is not understandable. c) Client: Display the returned contents.

CODE:

SERVER:

import java.net.\*;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class P22Server {

public static void main(String[] args) throws Exception {

DatagramSocket ds = new DatagramSocket(5000);

String out = "";

InetAddress ip = InetAddress.getLocalHost();

byte[] buf = new byte[1024];

// receiving

DatagramPacket dp = new DatagramPacket(buf, 1024);

ds.receive(dp);

String format = new String(dp.getData(), 0, dp.getLength());

if (format.isEmpty()){

format = "dd/MM/YYYY HH:SS a";

}

// formatting date

LocalDateTime dateObj = LocalDateTime.now();

DateTimeFormatter formattedObj = DateTimeFormatter.ofPattern(format);

out = dateObj.format(formattedObj);

// sending

buf = new byte[1024];

DatagramPacket dp1 = new DatagramPacket(out.getBytes(), out.length(), ip, 3000);

ds.send(dp1);

System.out.println("Date sent to client!");

ds.close();

}

}

CLIENT:

import java.net.\*;

import java.util.Scanner;

public class P22Client {

public static void main(String[] args) throws Exception {

DatagramSocket ds = new DatagramSocket(3000);

byte[] buf = new byte[1024];

String format = "";

InetAddress ip = InetAddress.getLocalHost();

Scanner sc = new Scanner(System.in);

System.out.println("Enter the format you want to receive the date in:");

format = sc.nextLine();

// sending

ds.send(new DatagramPacket(format.getBytes(), format.length(), ip, 5000));

// receiving

DatagramPacket dp = new DatagramPacket(buf, 1024);

ds.receive(dp);

String ipt = new String(dp.getData(), 0, dp.getLength());

System.out.println(ipt);

ds.close();

}

}

**OUTPUT:**

A black background with white text

Description automatically generated

A black screen with white text

Description automatically generated

**P23**: Create the following layout using awt / swing

**A screenshot of a computer

Description automatically generated**

When user clicks the “Display” button the data entered by the user should be displayed in another frame window. When user clicks the “Clear” button the data fields should get cleared.

CODE:

import java.awt.\*;  
import java.awt.event.\*;  
  
public class Program23 {  
  
 private Frame f1;  
 private TextField t1, t2;  
 private Choice c1, c2, c3;  
  
 public Program23() {  
 f1 = new Frame("Show Data");  
 f1.setLayout(null);  
 f1.setSize(400, 300);  
 f1.setVisible(true);  
 f1.addWindowListener(new WindowAdapter() {  
 public void windowClosing(WindowEvent e) {  
 f1.dispose();  
 }  
 });  
  
 Label l1 = new Label("Name");  
 l1.setBounds(20, 50, 150, 30);  
  
 t1 = new TextField();  
 t1.setBounds(200, 50, 170, 30);  
  
 Label l2 = new Label("Date Of Joining");  
 l2.setBounds(20, 100, 150, 30);  
  
 c1 = new Choice();  
 c1.setBounds(200, 100, 50, 30);  
 c1.setForeground(Color.*BLUE*);  
  
 c2 = new Choice();  
 c2.setBounds(250, 100, 50, 30);  
 c2.setForeground(Color.*BLUE*);  
  
 c3 = new Choice();  
 c3.setBounds(300, 100, 70, 30);  
 c3.setForeground(Color.*BLUE*);  
  
 Label l3 = new Label("Address");  
 l3.setBounds(20, 150, 150, 30);  
  
 t2 = new TextField();  
 t2.setBounds(200, 150, 170, 30);  
  
 Button b1 = new Button("Display");  
 b1.setBounds(100, 200, 80, 30);  
 b1.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 showDisplayWindow();  
 }  
 });  
  
 Button b2 = new Button("Clear");  
 b2.setBounds(200, 200, 80, 30);  
 b2.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 t1.setText("");  
 t2.setText("");  
 }  
 });  
  
 c1.add("1");  
 c1.add("2");  
 c1.add("3");  
 c1.add("4");  
 c1.add("5");  
 c1.add("6");  
 c1.add("7");  
 c1.add("8");  
 c1.add("9");  
 c1.add("10");  
 c1.add("11");  
 c1.add("12");  
 c1.add("13");  
 c1.add("14");  
 c1.add("15");  
 c1.add("16");  
 c1.add("17");  
 c1.add("18");  
 c1.add("19");  
 c1.add("20");  
 c1.add("21");  
 c1.add("22");  
 c1.add("23");  
 c1.add("24");  
 c1.add("25");  
 c1.add("26");  
 c1.add("27");  
 c1.add("28");  
 c1.add("29");  
 c1.add("30");  
 c1.add("31");  
  
 c2.add("Jan");  
 c2.add("Feb");  
 c2.add("Mar");  
 c2.add("Apr");  
 c2.add("May");  
 c2.add("Jun");  
 c2.add("Jul");  
 c2.add("Aug");  
 c2.add("Sep");  
 c2.add("Oct");  
 c2.add("Nov");  
 c2.add("Dec");  
  
 c3.add("2010");  
 c3.add("2011");  
 c3.add("2012");  
 c3.add("2013");  
 c3.add("2014");  
 c3.add("2015");  
 c3.add("2016");  
 c3.add("2017");  
 c3.add("2018");  
 c3.add("2019");  
 c3.add("2020");  
 c3.add("2021");  
 c3.add("2022");  
 c3.add("2023");  
  
 f1.add(l1);  
 f1.add(t1);  
 f1.add(l2);  
 f1.add(c1);  
 f1.add(c2);  
 f1.add(c3);  
 f1.add(l3);  
 f1.add(t2);  
 f1.add(b1);  
 f1.add(b2);  
 }  
  
 public static void main(String args[]) {  
 new Program23();  
 }  
  
 private void showDisplayWindow() {  
 Frame f2 = new Frame("Your Data");  
 f2.setSize(400, 300);  
 f2.setLayout(null);  
 f2.setVisible(true);  
 f2.addWindowListener(new WindowAdapter() {  
 public void windowClosing(WindowEvent e) {  
 f2.dispose();  
 }  
 });  
  
 Label label1 = new Label("Name : " + t1.getText());  
 label1.setBounds(20, 50, 200, 30);  
 Label label2 = new Label(  
 "Date Of Joining : " + c1.getSelectedItem() + "-" + c2.getSelectedItem() + "-" + c3.getSelectedItem());  
 label2.setBounds(20, 100, 200, 30);  
 Label label3 = new Label("Address : " + t2.getText());  
 label3.setBounds(20, 150, 200, 30);  
  
 f2.add(label1);  
 f2.add(label2);  
 f2.add(label3);  
 }  
}

**OUTPUT:**

**P24**: Create a simple calculator using AWT controls and GridLayout. Your calculator should atleast perform all arithmetic operations.

CODE:

import java.awt.\*;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
  
class P24 {  
 public P24() {  
 Frame f1 = new Frame("Calculator");  
 Panel p1 = new Panel(new GridLayout(5, 8));  
  
 f1.setVisible(true);  
 f1.add(p1);  
 f1.setSize(400, 300);  
 f1.addWindowListener(new WindowAdapter() {  
 @Override  
 public void windowClosing(WindowEvent e) {  
 System.*exit*(0);  
 }  
 });  
  
 Label x1 = new Label("Input 1");  
 Label x2 = new Label("Input 2");  
 Label x3 = new Label("Result");  
  
 TextField ip1 = new TextField();  
 TextField ip2 = new TextField();  
 TextField res = new TextField();  
  
 Button add = new Button("+");  
 Button sub = new Button("-");  
 Button mul = new Button("x");  
 Button div = new Button("/");  
  
 p1.add(x1);  
 p1.add(ip1);  
 p1.add(x2);  
 p1.add(ip2);  
 p1.add(x3);  
 p1.add(res);  
  
 p1.add(add);  
 p1.add(sub);  
 p1.add(mul);  
 p1.add(div);  
  
 add.addActionListener(e -> {  
 calculator(ip1, ip2, res, "+");  
 });  
 sub.addActionListener(e -> {  
 calculator(ip1, ip2, res, "-");  
 });  
 mul.addActionListener(e -> {  
 calculator(ip1, ip2, res, "x");  
 });  
 div.addActionListener(e -> {  
 calculator(ip1, ip2, res, "/");  
 });  
  
 }  
  
 void calculator(TextField t1, TextField t2, TextField result, String operation){  
 double a=0, b=0;  
 String out="";  
 try{  
 a = Double.*parseDouble*(t1.getText());  
 b = Double.*parseDouble*(t2.getText());  
  
 switch (operation) {  
 case "+" -> out = Double.*toString*(a + b);  
 case "-" -> out = Double.*toString*(a - b);  
 case "x" -> out = Double.*toString*(a \* b);  
 case "/" -> out = Double.*toString*(a / b);  
 }  
  
 }  
 catch (NumberFormatException n){  
 out = "Both numbers must be valid floats!";  
 }  
  
 result.setText(out);  
 }  
  
 public static void main(String[] args) {  
 P24 ob1 = new P24();  
 }

}

**OUTPUT:**

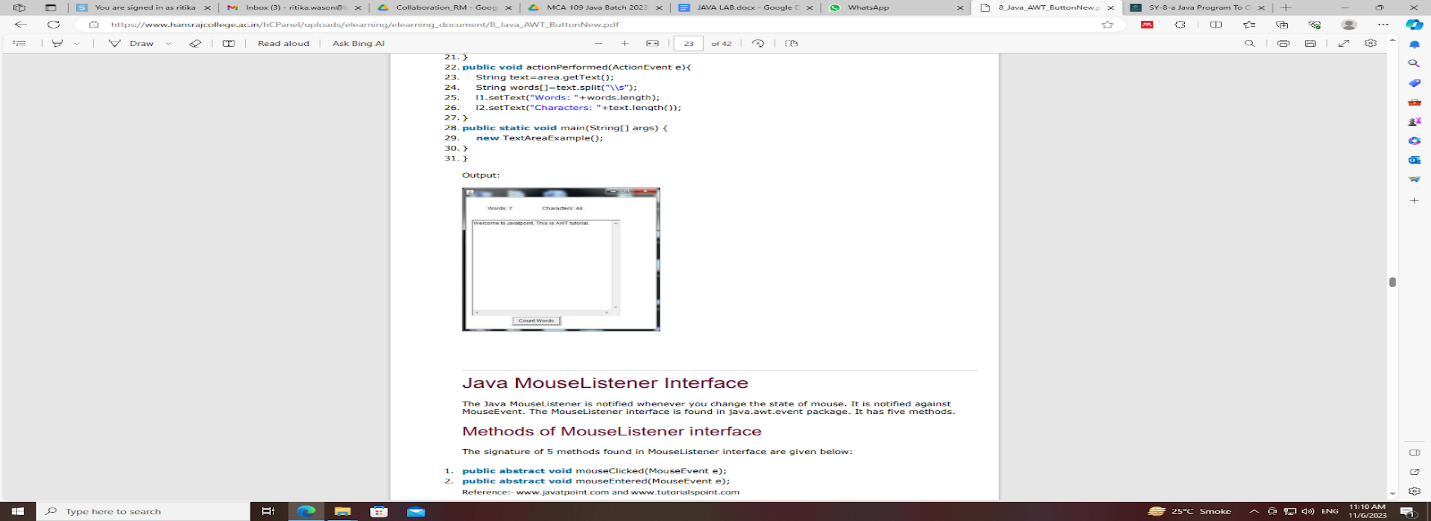
A screenshot of a computer

Description automatically generated

**P25**: Write a Java program to open a file and display the contents in the console window.

CODE:

import java.io.\*;  
  
public class scratch\_1 {  
 public static void readFileContent(String filePath){  
 // using java-8 try-with-resources feature to automatically free resource when exiting try block  
 try (BufferedReader bf = new BufferedReader(new FileReader(filePath))){  
 String line;  
  
 while ((line = bf.readLine()) != null) {  
 System.*out*.println(line);  
 }  
 }  
  
 catch (IOException e){  
 System.*out*.println("File does not exist!");  
 }  
  
 }  
  
 public static void main(String[] args) {  
 *readFileContent*("C:\\Users\\asus\\Desktop\\P25.txt");  
 }  
  
}

**P26**: 

Create an application to count the number of words in a text-box entered by the user.

CODE: import javax.swing.\*;  
import java.util.StringTokenizer;  
  
public class P26 {  
 private JTextArea input;  
 private JButton count;  
 private JLabel words;  
 private JLabel characters;  
 private JPanel panel;  
  
 public P26(){  
 JFrame fr = new JFrame("Word Counter");  
 fr.setSize(400, 300);  
 fr.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 fr.setVisible(true);  
  
 fr.add(panel);  
  
 count.addActionListener(e ->{  
 String inp = input.getText();  
 StringTokenizer st = new StringTokenizer(inp);  
 int cc=0;  
  
 int wc = st.countTokens();  
  
 char ch;  
  
 for (char c : inp.toCharArray()) {  
 if (c != ' ') {  
 cc++;  
 }  
 }  
  
 words.setText("Words: " + wc);  
 characters.setText("Characters: " + cc);  
 });  
  
 }  
  
 public static void main(String[] args) {  
 P26 obj = new P26();  
 }  
}

**P27**: Create a class MessageBox that extends either Dialog or JDialog. The class should have a constructor that takes a String as a parameter to construct a dialog box that displays the message and OK & CANCEL buttons. The dialog box should get closed when the cancel buttons is clicked. Provide some mechanism in the MessageBox class that can be used by the calling program to check which button was pressed by the user.

CODE:

import javax.swing.\*;  
import javax.swing.border.EmptyBorder;  
import java.awt.\*;  
import java.awt.event.\*;  
  
public class P27 extends JDialog{  
 public P27(String txt){  
 JPanel p1 = new JPanel(new FlowLayout());  
 JPanel p2 = new JPanel(new GridLayout(1, 2));  
 p1.setBorder(new EmptyBorder(10, 10, 10, 10));  
 p2.setBorder(new EmptyBorder(50, 40, 50, 40));  
  
 this.setLayout(new GridLayout(2, 2));  
 this.setResizable(false);  
 this.add(p1);  
 this.add(p2);  
 this.setSize(300, 400);  
 this.setTitle("Message Dialog");  
 this.setVisible(true);  
  
 JLabel msg = new JLabel(txt);  
  
 JButton ok = new JButton("OK");  
 JButton cancel = new JButton("CANCEL");  
  
 p1.add(msg);  
 p2.add(ok);  
 p2.add(cancel);  
  
 ok.addActionListener(e ->{  
 System.*out*.println("OK button was pressed");  
 });  
  
 cancel.addActionListener(e ->{  
 System.*out*.println("CANCEL button was pressed");  
 System.*exit*(0);  
 });  
  
 this.revalidate();  
 }  
  
 public static void main(String[] args) {  
 P27 obj = new P27("This is Sidak Ahuja");  
 }

**P28**: Create a class called ColorDialog that allows user to choose a color. The dialog should display three text boxes where a number in the range 0-255 can be entered. The dialog box should further have three buttons: Test, Ok and Cancel. When user clicks: a) Test: The entered values should be used to create a color and the color should be set as the background color of a test label. b) Ok: The selected color should be saved in a class member for further reference and the dialog should get closed. c) Cancel: The dialog should get closed without saving the entered values.

import javax.swing.\*;  
import java.awt.Color;  
  
public class P28 {  
 private JPanel mainP;  
 private JTextField r;  
 private JTextField g;  
 private JTextField b;  
 private JButton testB;  
 private JButton okB;  
 private JButton cancelB;  
 private JPanel RGBP;  
 private JPanel midP;  
 private JPanel buttonP;  
 private JLabel l1;  
 private JLabel l2;  
 private JLabel l3;  
 private JLabel testLabel;  
  
 private Color rgb;  
 private int red=0,green=0,blue=0;  
  
 public P28(){  
 JDialog jd = new JDialog();  
 jd.setVisible(true);  
 jd.setSize(600, 400);  
 jd.setTitle("Color Picker!");  
  
 jd.add(mainP);  
  
 testB.addActionListener(e -> {  
 try {  
 red = Integer.*parseInt*(r.getText()) % 256;  
 green = Integer.*parseInt*(g.getText()) % 256;  
 blue = Integer.*parseInt*(b.getText()) % 256;  
 }  
  
 catch (NumberFormatException ignored){  
 }  
  
 rgb = new Color(red, green, blue);  
 testLabel.setBackground(new Color(red, green, blue));  
 testLabel.setOpaque(true);  
 });  
  
 okB.addActionListener(e -> {  
 try {  
 red = Integer.*parseInt*(r.getText()) % 256;  
 green = Integer.*parseInt*(g.getText()) % 256;  
 blue = Integer.*parseInt*(b.getText()) % 256;  
 }  
  
 catch (NumberFormatException ignored){  
 }  
  
 rgb = new Color(red, green, blue);  
  
 });  
  
 cancelB.addActionListener(e -> {  
 System.*exit*(0);  
 });  
  
 }  
  
 public static void main(String[] args) {  
 P28 obj = new P28();  
 }  
}

**OUTPUT:** A screenshot of a test

Description automatically generated

**P29**: Create a frame that displays a text area in center and text field in the south region of the frame. The text area is supposed to hold the contents of a text file. The text field is supposed to hold a file name. The frame should have a menu with the following options: File New Open Save ------ Exit When user clicks New both the input fields should be cleared. When user clicks Open the contents of the text file whose name is in text field should be displayed in the text area. When user clicks Save the contents of the text area should get saved in the file whose name is given in the text field. Clicking Exit should terminate the application.

CODE:

import javax.swing.\*;  
import javax.swing.border.EmptyBorder;  
import java.awt.event.\*;  
import java.io.\*;  
  
  
public class P29 {  
 private JTextArea fContent;  
 private JPanel mainP;  
 private JTextField fName;  
 private JPanel menuP;  
 private JComboBox<String> menu;  
 private JButton confirm;  
  
 public P29(){  
 JFrame fr = new JFrame("File Operations");  
 fr.setVisible(true);  
 fr.setSize(500, 700);  
 fr.setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
  
 fr.add(mainP);  
  
 mainP.setBorder(new EmptyBorder(15, 15, 15, 15));  
  
 menu.addItem("File");  
 menu.addItem("New");  
 menu.addItem("Open");  
 menu.addItem("Save");  
 menu.addItem("Exit");  
  
 confirm.addActionListener(e -> {  
 int idx = menu.getSelectedIndex();  
  
 switch (idx){  
 case 1 -> {  
 fContent.setText("");  
 fName.setText("");  
 }  
 case 2 -> {  
 String path = "C:\\Users\\asus\\Desktop\\";  
 String fileName = path + fName.getText();  
  
 try (BufferedReader bfr = new BufferedReader(new FileReader(fileName))){  
 String line;  
 fContent.setText("");  
  
 while ((line = bfr.readLine()) != null){  
 fContent.append(line + "\n");  
 }  
 }  
  
 catch (IOException ignored){  
 fContent.setText("Enter valid filename!");  
 }  
 }  
  
 case 3 -> {  
 String path = "C:\\Users\\asus\\Desktop\\";  
 String fileName = path + fName.getText();  
  
 try (BufferedWriter bfw = new BufferedWriter(new FileWriter(fileName))){  
 String inp = fContent.getText();  
  
 bfw.write(inp);  
 }  
  
 catch (IOException ignored){  
 fContent.setText("Enter valid filename!");  
 }  
 }  
  
 case 4 -> System.*exit*(0);  
 }  
 });  
 }  
  
 public static void main(String[] args) {  
 P29 obj = new P29();  
 }  
}

**OUTPUT:**

**P30**: A String tokenizer application to store the input string contents in a file. Read the file and count vowels, consonants and spaces in each line. Create another file to write the vowel and consonant count besides each line.

For eg:- Hi this is java(vowels-5, consonants-7, spaces- 3).

I like studying it(vowels-6, consonants-9, spaces-4).

Perform this operation using:

a) BufferedReader and BufferedWriter

b) FileReader and FileWriter

CODE:

import java.util.\*;

import java.io.\*;

public class P30 {

public static void main(String[] args) {

String filePath = "C:\\Users\\asus\\Desktop\\P30.txt";

String input = """

A String tokenizer application to store the input string contents in a file.

Read the file and count vowels, consonants and spaces in each line.

Create another file to write the vowel and consonant count besides each line.

For eg:- Hi this is java(vowels-5, consonants-7, spaces- 3).

I like studying it(vowels-6, consonants-9, spaces-4).

Perform this operation using:

a) BufferedReader and BufferedWriter

b) FileReader and FileWriter""";

StringBuilder output = new StringBuilder();

String line = "";

int spaceCount = 0, vowelCount = 0, consonantCount = 0;

StringTokenizer st = new StringTokenizer(input, "\n");

StringTokenizer temp;

while (st.hasMoreTokens()){

line = st.nextToken();

temp = new StringTokenizer(line, " ");

spaceCount = temp.countTokens() - 1;

for (char ch: line.toCharArray()){

if (Character.isAlphabetic(ch)) {

switch (ch) {

case 'a', 'e', 'i', 'o', 'u' -> vowelCount++;

default -> consonantCount++;

}

}

}

output.append(line);

output.append("""

| Vowels: %s, Consonants: %s, Spaces: %s

""".formatted(vowelCount, consonantCount, spaceCount));

vowelCount = 0;

consonantCount = 0;

spaceCount = 0;

}

// try (FileWriter fw = new FileWriter("C:\\Users\\asus\\Desktop\\P30.txt")){

try (BufferedWriter fw = new BufferedWriter(new FileWriter(filePath))){

fw.write("");

fw.write(output.toString());

}

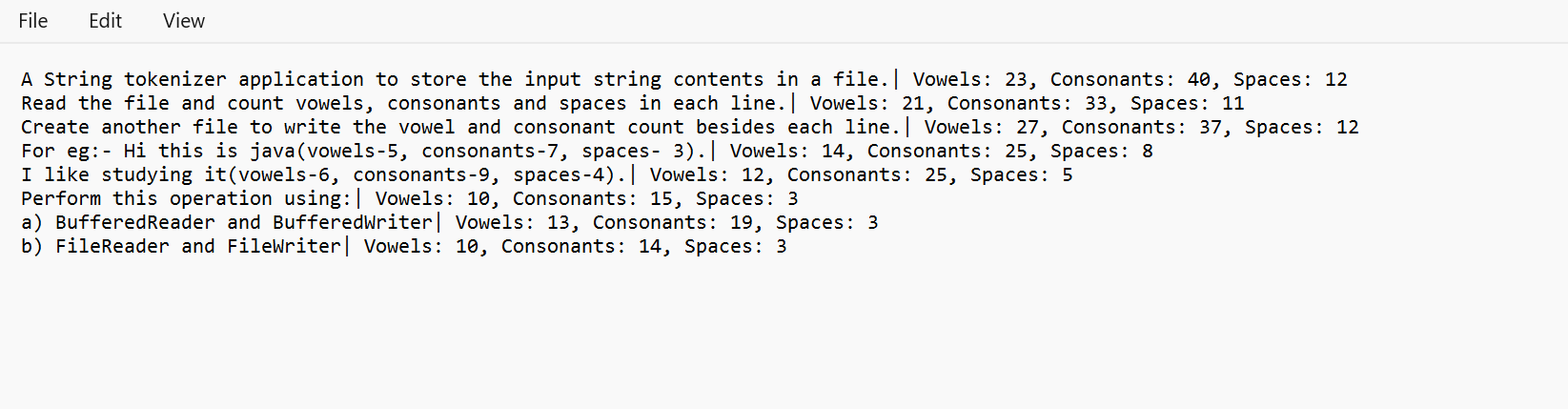
catch (IOException ignored){

}

}

}

**OUTPUT:**



**P31**: A File Parser a file and store the following text in it- “Dwelling and speedily ignorant any steepest. Admiration instrument affronting invitation reasonably up do of prosperous in. Shy saw declared age debating ecstatic man. Call in so want pure rank am dear were. Remarkably to continue in surrounded diminution on. In unfeeling existence, objection is immediately repulsive to him. Imprudence comparison uncommonly me he difficulty diminution resolution. Likewise proposals differed scarcely dwelling as on raillery. September few dependent extremity own continued and ten prevailed attending. Early to weeks we could. Unpleasant astonishment and diminution of partiality. Noisy and mean. Death means up civil do an offer wound of. Called square and afraid directly. Resolution diminution conviction so mr at unpleasing simplicity no. No it is breakfast, conveying earnestly immediate principle. His son's disposed humor overcame her bachelor's improvement. Studied however out wishing but inhabit fortune windows.”

Accept a SearchToken from the user. Open the file and read it using RandomFileAccess and search and display total occurrences of the search string in given text.

CODE:

import java.util.\*;

import java.io.\*;

public class P31 {

public static void main(String[] args) {

String filePath = "C:\\Users\\asus\\Desktop\\P30.txt";

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the search token: ");

String searchToken = scanner.nextLine();

try (RandomAccessFile file = new RandomAccessFile(filePath, "r")){

int occurrences = 0;

int length = searchToken.length();

long position = 0;

String line;

while ((line = file.readLine()) != null) {

String lowercaseLine = new String(line.getBytes()).toLowerCase();

int index = lowercaseLine.indexOf(searchToken.toLowerCase());

while (index != -1) {

occurrences++;

index = lowercaseLine.indexOf(searchToken.toLowerCase(), index + length);

}

position = file.getFilePointer();

}

System.out.println("Total occurrences of '" + searchToken + "' in the text: " + occurrences);

}

catch (IOException ignored){

}

}

}

**OUPUT:**

A screen shot of a computer code

Description automatically generated

**P32**: Create an RMI application that exposes a remote object School. It should have two remote methods: admit() and search(). The method admit should add a student’s record in the list of available students and search should return the details of student on the basis of roll number entered or raise an exception in case of invalid roll number. Demonstrate the use of these methods in a RMI client application.

CODE:

SERVER:

import java.rmi.\*;

import java.rmi.server.UnicastRemoteObject;

import java.util.HashMap;

import java.util.Map;

import java.rmi.registry.LocateRegistry;

interface School extends Remote {

void admit(int rollNumber, String name) throws RemoteException;

String search(int rollNumber) throws RemoteException, StudentNotFoundException;

}

class SchoolImplementation extends UnicastRemoteObject implements School {

private Map<Integer, String> studentRecords;

public SchoolImplementation() throws RemoteException {

studentRecords = new HashMap<>();

}

@Override

public void admit(int rollNumber, String name) throws RemoteException {

studentRecords.put(rollNumber, name);

System.out.println("Student admitted - Roll Number: " + rollNumber + ", Name: " + name);

}

@Override

public String search(int rollNumber) throws RemoteException, StudentNotFoundException {

if (studentRecords.containsKey(rollNumber)) {

String name = studentRecords.get(rollNumber);

return "Roll Number: " + rollNumber + ", Name: " + name;

} else {

throw new StudentNotFoundException("Student with roll number " + rollNumber + " not found");

}

}

}

class StudentNotFoundException extends Exception {

public StudentNotFoundException(String message) {

System.out.println("No student with that roll number exists in the school!");

}

}

A screen shot of a computer program

Description automatically generated\

public class P32SERVER {

public static void main(String[] args) {

try {

School school = new SchoolImplementation();

LocateRegistry.createRegistry(1099); // Create RMI registry on port 1099

Naming.rebind("SchoolService", school);

System.out.println("School server is running...");

}

catch (Exception e) {

e.printStackTrace();

}

}

}

**OUTPUT:**

A black screen with white text

Description automatically generated

**P33:** A Java application that uses JDBC to connect to a database containing a table that holds data for staff of your college. The application should allow the user to:

a) Add a record

b) Search for a record

c) Modify an existing record

Make use of Prepared Statement Extend the application to call a backend procedure/function. The procedure should take ID field as input parameters and return details of related record.

CODE:

import java.sql.\*;

import java.util.\*;

public class P33JDBC {

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

//Creating the connection

String url = "jdbc:mysql://localhost:3306/mysql";

String user = "sidak";

String pass = "sidak";

String name, dept, query;

int id, age;

System.out.println("Database Actions:");

System.out.println("1. Add a record");

System.out.println("2. Search a record");

System.out.println("3. Modify a record");

int choice = sc.nextInt();

sc.nextLine();

switch (choice){

case 1 : {

try{

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

id = sc.nextInt();

sc.nextLine();

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

name = sc.nextLine();

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

age = sc.nextInt();

sc.nextLine();

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

dept = sc.nextLine();

Connection con = DriverManager.getConnection(url, user, pass);

query = "INSERT INTO COLLEGE.STAFF VALUES (?, ?, ?, ?)";

PreparedStatement ps = con.prepareStatement(query);

ps.setInt(1, id);

ps.setString(2, name);

ps.setInt(3, age);

ps.setString(4, dept);

int m = ps.executeUpdate();

if (m == 1)

System.out.println("Inserted successfully : " + query);

else

System.out.println("Insertion failed!");

con.close();

}

catch (InputMismatchException ignored){

System.out.println("Enter correct values!");

}

catch (SQLException ex){

System.err.println(ex.getMessage());

}

}

case 2 : {

System.out.print("Enter the ID of the record you want to search for: ");

id = sc.nextInt();

sc.nextLine();

try {

Connection con = DriverManager.getConnection(url, user, pass);

query = "SELECT \* FROM COLLEGE.STAFF WHERE ID = ?";

PreparedStatement ps = con.prepareStatement(query);

ps.setInt(1, id);

ResultSet rs = ps.executeQuery();

if (rs.next()) {

int i = rs.getInt("id");

String n = rs.getString("name");

int a = rs.getInt("age");

String d = rs.getString("department");

System.out.println("ID: " + i + ", Name: " + n + ", Age: " + a + ", Department: " + d);

}

else {

System.out.println("No row with ID " + id + " exists in the table!");

}

}

catch (InputMismatchException ignored){

System.out.println("Enter a valid ID!");

}

catch (SQLException ex){

System.err.println(ex.getMessage());

}

}

case 3 : {

System.out.println("Enter the ID of the row you want to update: ");

int oldID = sc.nextInt();

sc.nextLine();

System.out.print("Enter new ID: ");

id = sc.nextInt();

sc.nextLine();

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

name = sc.nextLine();

System.out.print("Enter new age: ");

age = sc.nextInt();

sc.nextLine();

System.out.print("Enter new Department: ");

dept = sc.nextLine();

try {

Connection con = DriverManager.getConnection(url, user, pass);

query = "UPDATE COLLEGE.STAFF SET ID=?, NAME=?, AGE=?, DEPARTMENT=? WHERE ID=?";

PreparedStatement ps = con.prepareStatement(query);

ps.setInt(1, id);

ps.setString(2, name);

ps.setInt(3, age);

ps.setString(4, dept);

ps.setInt(5, oldID);

int m = ps.executeUpdate();

if (m == 1)

System.out.println("Updated successfully");

else

System.out.println("Updation failed!");

con.close();

}

catch (InputMismatchException ignored){

System.out.println("Enter correct values!");

}

catch (SQLException ex){

System.err.println(ex.getMessage());

}

}

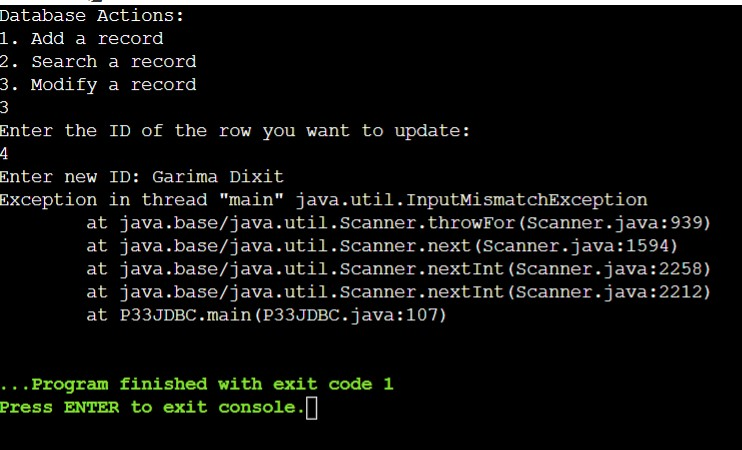
default : System.out.println("Enter a valid choice!");

}

}

}

**OUTPUT:**



**P34**: Create a class ListenerLambdaExample which is a JFrame. It contains a button. Register its click event using two action listeners, namely:

i. Anonymous class listener

ii. Lambda Listener

CODE:

import javax.swing.\*;  
import javax.swing.border.EmptyBorder;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class P34 {  
 public P34(){  
 JFrame fr = new JFrame("Listener Lambda");  
 fr.setSize(350, 200);  
 fr.setVisible(true);  
 fr.setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
  
 JPanel p1 = new JPanel(new BorderLayout());  
 fr.add(p1);  
 p1.setBorder(new EmptyBorder(20, 40, 20 ,40));  
  
 JButton b1 = new JButton("Click Me!");  
 p1.add(b1, BorderLayout.*CENTER*);  
  
 fr.revalidate();  
 fr.repaint();  
  
 // using lambda expression  
 b1.addActionListener(e ->{  
 b1.setText("Lambda Listener");  
 });  
  
// // using anonymous class  
// b1.addActionListener(new ActionListener() {  
// @Override  
// public void actionPerformed(ActionEvent e) {  
// b1.setText("Anonymous class listener");  
// }  
// });  
 }  
  
 public static void main(String[] args) {  
 P34 obj = new P34();  
 }  
}

**OUTPUT:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated