//Index No. 200323V

import java.io.\*;  
import java.sql.SQLOutput;  
import java.time.LocalDate;  
import java.time.format.DateTimeFormatter;  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.util.Date;  
import java.text.SimpleDateFormat;  
  
public class Email\_Client {  
 public static void main(String[] args) {  
 // getting the todays date  
 LocalDate TodayDate = LocalDate.*now*();  
  
 //creating instannces of all the recipients stored in the memory.  
 ArrayList<Recipient> Recipients = *CreateRecipientObjects*("clientList.txt");  
 ArrayList<Email> SentEmails = *Deserialize*();  
  
 //bday emails sent today  
 ArrayList<Email> sentBdayEmailsToday = new ArrayList<Email>();  
 for (Email E: SentEmails){  
 if ((E.getSubject().equalsIgnoreCase("Birthday Wish") && (E.getDate().equals(TodayDate) ))){  
 sentBdayEmailsToday.add(E);  
 }  
 }  
 //sending birthday wish getEmail()  
 ArrayList<Recipient> RecipientsHavingBirthday = new ArrayList<Recipient>();  
 //Getting all the recipients having birthdays in to arraylist  
 if(sentBdayEmailsToday.isEmpty()) {  
 for (Recipient R: Recipients) {  
 if (R instanceof Personal) {  
 Personal per = (Personal) R;  
 if (per.getBirthday().getDayOfYear() == TodayDate.getDayOfYear()) {  
 Email bdayemail = new Email(per, "hugs and love on your birthday", "Birthday Wish");  
 bdayemail.send();  
 *saveOnHard*(bdayemail);  
 }  
 }  
  
 if (R instanceof Official\_Friend) {  
 Official\_Friend Off = (Official\_Friend) R;  
 if (Off.getBirthday().getDayOfYear() == TodayDate.getDayOfYear()) {  
 Email bdayemail = new Email(Off, "many happy returns of the day", "Birthday Wish");  
 bdayemail.send();  
 *saveOnHard*(bdayemail);  
 }  
 }  
 }  
 }  
 System.*out*.println("Enter option type\n"  
 + "1 - Adding a new recipient\n"  
 + "2 - Sending an email\n"  
 + "3 - Printing out all the recipients who have birthdays\n"  
 + "4 - Printing out details of all the emails sent\n"  
 + "5 - Printing out the number of recipient objects in the application");  
  
  
  
 //getting the user input  
 Scanner scanner = new Scanner(System.*in*);  
 int option = scanner.nextInt();  
  
 switch( option){  
 case 1 : // Adding new recipient  
 System.*out*.println("Enter details of the new recipient in the following order\n"  
 + "If an official recipient \t official: <name>,<email>,<designation>\n"  
 + "if an office friend recipient\t Office\_Friend: <name>,<email>,<designation>,<birthday>\n"  
 + "if a personal friend \t Personal: <name>,<nick-name>,<email>,birthday\n");  
  
 Scanner inputScanner1 = new Scanner(System.*in*);  
 String input1 = inputScanner1.nextLine(); // getting the input string  
  
 try {//code to add a new member to the client list  
 BufferedWriter writer = new BufferedWriter(new FileWriter("clientList.txt",true));  
 writer.write(input1);  
 writer.write("\n");  
 writer.close();  
  
 //add this newly entered recipient to the Recipient Arraylist  
 Recipient newlyAddedReci = *CreateObject*(input1);  
 Recipients.add(newlyAddedReci);  
  
// DateTimeFormatter fomatter2 = DateTimeFormatter.ofPattern("yyyy/MM/dd");  
// this.Birthday = LocalDate.parse(Birthday,fomatter2);  
  
 if (newlyAddedReci instanceof Personal){  
 if (LocalDate.*parse*(input1.strip().split(",")[3],DateTimeFormatter.*ofPattern*("yyyy/MM/dd")).isEqual(((Personal)newlyAddedReci).getBirthday())) {  
 Email Em = new Email(newlyAddedReci, "many happy returns of the day", "Birthday Wish") ;  
 Em.send();  
 *saveOnHard*(Em);  
 }  
 }  
  
 if (newlyAddedReci instanceof Official\_Friend ){  
 if (LocalDate.*parse*(input1.strip().split(",")[3],DateTimeFormatter.*ofPattern*("yyyy/MM/dd")).isEqual(((Official\_Friend)newlyAddedReci).getBirthday())) {  
 Email Em = new Email(newlyAddedReci, "many happy returns of the day", "Birthday Wish") ;  
 Em.send();  
 *saveOnHard*(Em);  
 }  
 }  
 }  
 catch (IOException err){  
 err.printStackTrace();  
 }  
  
  
 break;  
  
 case 2:  
 System.*out*.println("Enter the details of the email want to send in the following order\n"  
 +"<email>,<subject>,<content>");  
  
 Scanner inputScanner2 = new Scanner(System.*in*);  
 String input2 = inputScanner2.nextLine(); // getting the input string  
  
 String[] emailDetail = input2.split(",",-2);  
 for(int i = 0; i < Recipients.size(); i++){  
 System.*out*.println("iiiiiiiiiii");  
 // searching for the Recipient object having the same email address  
 if (Recipients.get(i).getEmail().equals(emailDetail[0])) {  
 System.*out*.println("ttttt");  
 Email email1 = new Email(Recipients.get(i), emailDetail[1], emailDetail[2]);  
 email1.send();  
 *saveOnHard*(email1);  
 break;  
 }  
 }  
  
  
  
 break;  
  
  
 case 3:  
 // birthday  
 // input format year/month/day  
 System.*out*.println("Enter the date in the following order \nyear/month/day");  
  
 Scanner inputScanner3 = new Scanner(System.*in*);  
 String inputDate = inputScanner3.nextLine(); //reading the user input  
  
 //convert the inputdate into LocalDate  
 DateTimeFormatter fomatter = DateTimeFormatter.*ofPattern*("yyyy/MM/dd");  
 LocalDate TheDate = LocalDate.*parse*(inputDate,fomatter);  
  
 //ArrayList<Recipient> RecipientsSet = CreateRecipientObjects("clientList.txt"); // create recipient objects  
  
 for (Recipient TempReci:Recipients){  
 //as only personal and official\_friends has birthdays, no need to check Officials  
 if (TempReci instanceof Personal) {  
 Personal per = (Personal) TempReci;  
 if (per.getBirthday().getDayOfYear() == TheDate.getDayOfYear()) {  
 System.*out*.println(per.getName());  
 }  
 }  
  
 if (TempReci instanceof Official\_Friend) {  
 Official\_Friend Off = (Official\_Friend) TempReci;  
 if (Off.getBirthday().getDayOfYear() == TheDate.getDayOfYear()) {  
 System.*out*.println(Off.getName());  
 }  
 }  
 }  
 break;  
  
  
 case 4:  
 //sent emails  
 System.*out*.println("Enter the date of the emails sent in the following order \nyear/month/day");  
  
 //getting the date input  
 Scanner inputScanner4 = new Scanner(System.*in*);  
 String dateInStr = inputScanner4.nextLine(); // getting the input string  
 LocalDate DateIn ;  
 DateTimeFormatter fomatter1 = DateTimeFormatter.*ofPattern*("yyyy/MM/dd");  
 DateIn = LocalDate.*parse*(dateInStr,fomatter1);  
  
  
  
  
 ArrayList<Email> AllTheEmailsSent = *Deserialize*();//getting all the emails sent  
  
 for (Email tempEmail : AllTheEmailsSent){  
 //if the sent date of the tempEmail is Today  
 LocalDate EmailDate = tempEmail.getDate();  
  
 //String EmailDate = tempEmail.getDate().toString().split("-")[0] +"/"+tempEmail.getDate().toString().split("-")[1]+"/"+tempEmail.getDate().toString().split("-")[2] ;  
 if (EmailDate.isEqual(DateIn)){  
 System.*out*.println("Reciever :\t"+tempEmail.getReceiversName()+",\t\tSubject :\t"+ tempEmail.getSubject());  
 }  
 }  
 break;  
 case 5:  
 System.*out*.println(Recipients.size()); // printing the size of the ArrayList  
 break;  
 }  
  
 }  
 public static Recipient CreateObject(String S) {  
 // create an object getting the string  
 String[] A = S.trim().split(":");  
 String RecipientType = A[0].trim();  
 String[] Detail = A[1].split(",");  
  
 //finding the type of the recipient  
 if (RecipientType.equals("Personal")) {  
 Recipient r = new Personal(Detail[0], Detail[1], Detail[2], Detail[3]);  
 return r;  
 }  
 else if (RecipientType.equals("Official")) {  
 Recipient r = new Official(Detail[0], Detail[1], Detail[2]);  
 return r;  
 }  
 else if(RecipientType.equals("Official\_Friend")) {  
 Recipient r = new Official\_Friend(Detail[0], Detail[1], Detail[2], Detail[3]);  
 return r;  
 }  
 return null;  
 }  
 public static ArrayList<Recipient> CreateRecipientObjects(String FileName) {  
 // Return ArrayList containing all Recipient objects stored in the FileName  
 ArrayList<Recipient> RecipientArr = new ArrayList<>();  
 try {  
 File RecipientFile = new File("clientList.txt");  
 Scanner scan = new Scanner(RecipientFile);  
 while (scan.hasNextLine() ){  
 //read line by line  
 String line = scan.nextLine();  
 Recipient Res\_Object = *CreateObject*(line); // create Recipient object  
 // add Res\_Object object to the RecipientArr  
 RecipientArr.add(Res\_Object);  
 }  
 } catch (IOException error) {  
 error.printStackTrace();  
 }  
 return RecipientArr;  
 }  
  
 //to save the emails in the hard as a text file  
 public static void saveOnHard(Email emailObj){  
 ArrayList<Email> tempEmailList = *Deserialize*();  
 tempEmailList.add(emailObj);  
 *SerializeTheArr*(tempEmailList);  
 }  
  
  
 public static void SerializeTheArr(ArrayList<Email> Arr){  
 try {  
 // creating a outputStream to store the Arraylist object  
 ObjectOutputStream obj = new ObjectOutputStream( new FileOutputStream("sentEmails.txt"));  
 obj.writeObject(Arr); // serializing the Arraylist in the text file  
 obj.flush();  
 obj.close();  
 System.*out*.println("massage saved successfully");  
 }  
 //handling exceptions  
 catch (Exception err){  
 err.printStackTrace();  
 }  
 }  
  
 @SuppressWarnings("unchecked")  
 public static ArrayList<Email> Deserialize(){  
 // this method is to read the serialiezed Arraylist object  
 ArrayList<Email> emailList = new ArrayList<Email>();  
  
 try {  
 //reading from the text file  
 FileInputStream FileIn = new FileInputStream("sentEmails.txt");  
 ObjectInputStream ObjectIn = new ObjectInputStream(FileIn);  
 emailList = (ArrayList<Email>)ObjectIn.readObject();  
  
 ObjectIn.close();  
 FileIn.close();  
 }  
 //handling exceptions  
 catch (EOFException e){  
  
 }  
 catch (IOException io\_err){  
 io\_err.printStackTrace();  
 }  
 catch (ClassNotFoundException Class\_not\_found\_err){  
 Class\_not\_found\_err.printStackTrace();  
 }  
 return emailList ;  
}  
}

import java.io.Serializable;  
  
public abstract class Recipient implements Serializable {  
 private String Name;  
 private String Email;  
  
 public Recipient(String Name, String Email){  
 this.Email = Email;  
 this.Name = Name;  
 }  
 public String getName(){  
 return Name;  
 }  
 public String getEmail(){  
 return Email;  
 }  
  
}

import java.time.DateTimeException;  
import java.time.LocalDate;  
import java.time.format.DateTimeFormatter;  
import java.time.format.DateTimeParseException;  
  
public class Personal extends Recipient {  
 private String NickName;  
 public LocalDate Birthday;  
  
 public Personal(String Name, String NickName , String Email, String Birthday){  
 super(Name,Email);  
  
 DateTimeFormatter fomatter = DateTimeFormatter.*ofPattern*("yyyy/MM/dd");  
 this.Birthday = LocalDate.*parse*(Birthday,fomatter);  
 this.NickName = NickName;  
 }  
 public String getNickName(){  
 return NickName;  
 }  
  
 public LocalDate getBirthday(){  
 return Birthday;  
 }  
  
}

import java.time.LocalDate;  
import java.time.format.DateTimeFormatter;  
  
public class Official\_Friend extends Official {  
 public LocalDate Birthday;  
  
 public Official\_Friend(String Name, String Email ,String designation , String Birthday){  
 super(Name, Email, designation);  
 DateTimeFormatter fomatter = DateTimeFormatter.*ofPattern*("yyyy/MM/dd");  
 this.Birthday = LocalDate.*parse*(Birthday,fomatter);  
 }  
 public LocalDate getBirthday(){  
 return Birthday;  
 }  
  
  
}

public class Official extends Recipient{  
 private String designation;  
  
 public Official( String Name, String Email ,String designation){  
 super(Name,Email);  
 this.designation = designation;  
 }  
 public String getDesignation(){  
 return designation;  
 }  
  
}

import javax.mail.\*;  
import javax.mail.internet.InternetAddress;  
import javax.mail.internet.MimeMessage;  
import java.io.\*;  
import java.text.SimpleDateFormat;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.Properties;  
import javax.activation.\*;  
import java.util.Scanner;  
  
public class Email implements Serializable {  
 private Recipient toWhome;  
 private String Content;  
 private String Subject;  
 private LocalDate Date;  
  
 public String getReceiversName() {  
 return toWhome.getName();  
 }  
  
 public String getTheReceiversEmailAddress(){  
 return toWhome.getName();  
 }  
  
 public String getContent() {  
 return Content;  
 }  
  
 public String getSubject() {  
 return Subject;  
 }  
 public LocalDate getDate(){  
 return Date;  
 }  
 public Recipient getToWhome(){  
 return toWhome;  
 }  
  
  
 public Email(Recipient toWhome, String content, String subject) {  
 this.toWhome = toWhome;  
 this.Content = content;  
 this.Subject = subject;  
 LocalDate thisdate = LocalDate.*now*();  
 this.Date = thisdate;  
  
 //setting the addressing of the email for different recipient types  
 if (toWhome instanceof Personal) {  
 //if sending emails to a personal  
 String NickName = ((Personal) toWhome).getNickName();  
 LocalDate Birthday = ((Personal) toWhome).getBirthday();  
 // modifying the content of the email  
 if (this.Subject.equals( "Birthday Wish")){  
 this.Content = " Happy Birthday for turning " + String.*valueOf*(LocalDate.*now*().getYear()-Birthday.getYear()) +" years, "+ this.Content;  
 }  
 this.Content = NickName + ",\n" + this.Content;  
 }  
 else if (toWhome instanceof Official\_Friend) {  
 LocalDate Birthday = ((Official\_Friend) toWhome).getBirthday();  
 String Designation = ((Official\_Friend) toWhome).getDesignation();  
 //modifying the content  
 if (this.Subject.equals( "Birthday Wish")){  
 this.Content = " Happy Birthday for turning " + String.*valueOf*(LocalDate.*now*().getYear()-Birthday.getYear()) +" years and "+ this.Content;  
 }  
 this.Content = "Mr/Mrs/Miss " + toWhome.getName() + ",\n" + Designation + ".\n" + this.Content;  
  
 }  
 else if (toWhome instanceof Official) {  
 String Designation = ((Official) toWhome).getDesignation();  
 // modifying the content  
 this.Content = "Mr/Mrs/Miss " + toWhome.getName() + ",\n" + Designation + ".\n" + this.Content;  
  
 }  
  
 }  
  
  
 public void send() {  
 // senders details  
  
 String username = "kkajskumarasinghe@gmail.com";  
 String password = "ohrisgkqwlshpwpf";  
 //getting receivers details  
 String receiverName = toWhome.getName();  
 String receiverEmail = toWhome.getEmail();  
  
 Properties prop = new Properties();  
 prop.put("mail.smtp.host", "smtp.gmail.com");  
 prop.put("mail.smtp.port", "587");  
 prop.put("mail.smtp.auth", "true");  
 prop.put("mail.smtp.starttls.enable", "true"); //TLS  
 Session session = Session.*getInstance*(prop,  
 new javax.mail.Authenticator() {  
 protected PasswordAuthentication getPasswordAuthentication() {  
 return new PasswordAuthentication(username, password);  
 }  
 });  
 try {  
 Message message = new MimeMessage(session);  
 message.setFrom(new InternetAddress(username));  
 message.setRecipients(  
 Message.RecipientType.*TO*,  
 InternetAddress.*parse*(receiverEmail)  
 );  
 message.setSubject(Subject);  
 message.setText(Content);  
  
 Transport.*send*(message);  
  
 System.*out*.println("Massage sent!");  
  
 } catch (MessagingException e) {  
 e.printStackTrace();  
 }  
  
 }  
}