```
// your index number
// 200327L

import java.io.*;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.Scanner;
import javax.mail.*;

public class Email_Client {

   /**
    * email client have a file object(clientList.txt)
    * email client should maintain each email recipient object when running
    * A list of recipients to whom a birthday greeting should be sent is maintained in the application.
    **/
   static String name = "Thilakna";
   static File file = RecipientDataTextFileOperations.getInstance();
   static File serfile = SerializingAndDeserializingOperations.getInstance();

   // static array for store recipients
   static ArrayList<Recipient> recipients = new ArrayList<>();

   // static array for email_receivers
   static ArrayList<EmailReceiver> birthdayGreetingReceivers = new ArrayList<>();

   public static void main(String[] args) throws IOException, MessagingException,
ClassNotFoundException {
```

```
// maintain each email recipient in the application
RecipientDataTextFileOperations.load_recipients(recipients);

//maintain each birthday greeting receiver in the application
RecipientDataTextFileOperations.load_greetings_receivers(birthdayGreetingReceivers);

// sent wishes and save the emails
Email_Client.sendBirthdayWishes(birthdayGreetingReceivers);

Scanner scanner = new Scanner(System.in);

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");

int option = scanner.nextInt();

scanner.nextLine();

switch(option){
   case 1:

      String details = scanner.nextLine();

      //check for validity of the input
      String[] check01 = details.split(":");
```

```
    String[] check02 = details.split(",");
    if(check01.length ==2 && check02.length>=3 && check02.length<=4){
        // writing the details in to the file
        RecipientDataTextFileOperations.write(details);
    }
    else
        System.out.println("Invalid input!\nTry again!");



    break;
case 2:
    try {
        // input format - email, subject, content
        String input = scanner.nextLine();
        String[] input_list = input.split(",");
        String recipient = input_list[0];
        String subject =  input_list[1];
        String content = input_list[2];

        // code to send an email and it will be serialized
        EmailOperations.send(recipient,subject,content);
    }catch (ArrayIndexOutOfBoundsException e){
        System.out.println("Invalid input!\nTry again!");
    }



    break;
case 3:
```

```
    // input format - yyyy/MM/dd (ex: 2018/09/17)
    String inputForCase3= scanner.nextLine();
    try {
        // check the validity of the input
        String list_3[] = inputForCase3.split("/");
        //String Y = list_3[0];
        String M = list_3[1];
        String D = list_3[2];

        // code to print recipients who have birthdays on the given date
        Email_Client.findBirthdays(M,D);
    }catch (ArrayIndexOutOfBoundsException ex){
        System.out.println("Enter an valid input!");
        break;
    }
    break;
case 4:

    // input format - yyyy/MM/dd (ex: 2018/09/17)
    // code to print the details of all the emails sent on the input date
    try {
        String inputForCase4= scanner.nextLine();
        SerializingAndDeserializingOperations.deserializeAndPrint(serfile,inputForCase4);
    }catch (ArrayIndexOutOfBoundsException ex) {
        System.out.println("Enter an valid input!");
        break;
    }
    catch (Exception e){
```

```
                System.out.println("problem occurred while deserializing!");
                break;
            }
            break;
        case 5:

            // code to print the number of recipient objects in the application
            System.out.println(recipients.size());
            break;

    }

}

/**
 * this method sends birthday wishes by traversing the greetings receiver array
 * and selecting the recipients whom a birthday greeting should be sent
 **/
private static void sendBirthdayWishes(ArrayList<EmailReceiver> birthdayGreetingReceivers) throws
IOException, MessagingException {
    // get the current date
    LocalDate date = LocalDate.now();
    String[] YY_MM_DD = date.toString().split("-");
    String month = YY_MM_DD[1];
    String today = YY_MM_DD[2];

    // find the recipients who have birthday today
    for(int i=0; i<birthdayGreetingReceivers.size();i++){
        String birthday = null;
```

```
        String type = ((Recipient)birthdayGreetingReceivers.get(i)).type;
        if(type.equals("Office_friend"))
        {
           Office_friend Office_friend = (Office_friend)(birthdayGreetingReceivers.get(i));
           birthday = Office_friend.getBirthday();

        }
        else if(type.equals("Personal"))
        {
           Personal personal = (Personal)(birthdayGreetingReceivers.get(i));
           birthday = personal.getBirthday();

        }
        String[] birth_date = birthday.split("/");
        String B_Month = birth_date[1];
        String B_date = birth_date[2];

        // compare with today
        if(B_date.equals(today )&& B_Month.equals(month)){
           EmailOperations.sendGreetings(birthdayGreetingReceivers.get(i),type,name);
        }
     }


}

/**
 * this method is used to print all the recipients whom birthday is on a given day
```

```
 **/
public static void findBirthdays(String M, String D ){
   for(int i=0; i<birthdayGreetingReceivers.size();i++){
      String birthday = null;

      String type = ((Recipient)birthdayGreetingReceivers.get(i)).type;
      if(type.equals("Office_friend"))
      {
         Office_friend Office_friend = (Office_friend)(birthdayGreetingReceivers.get(i));
         birthday = Office_friend.getBirthday();
         String[] birth_date = birthday.split("/");
         String B_Month = birth_date[1];
         String B_date = birth_date[2];
         if(B_date.equals(D)&& B_Month.equals(M)){
            System.out.println(Office_friend.getName());
         }

      }
      else if(type.equals("Personal"))
      {
         Personal personal = (Personal)(birthdayGreetingReceivers.get(i));
         birthday = personal.getBirthday();
         String[] birth_date = birthday.split("/");
         String B_Month = birth_date[1];
         String B_date = birth_date[2];
         if(B_date.equals(D)&& B_Month.equals(M)){
            System.out.println(personal.getName());
         }
```

```
        }
      }
   }

}
//=============================================================================
=================================================================
//=============================================================================
=================================================================
abstract class Recipient {
   protected String type;
   protected String name;
   protected String email;
   protected static int number_of_recipients;

   protected Recipient() {

   }

   public String getEmail() {
      return email;
   }

   public String getName() {
      return name;
   }
}
//=============================================================================
=================================================================
```

```
//============================================================================
============================================================

interface EmailReceiver {
   public String getBirthday();
}

//============================================================================
============================================================
//============================================================================
============================================================
class Personal extends Recipient implements  EmailReceiver{
   private final String birthday;
   private final String nickName;
   public Personal(String[] arr) {
      super();
      this.type = "Personal";
      this.name = arr[0];
      this.nickName = arr[1];
      this.email = arr[2];
      this.birthday = arr[3];
      number_of_recipients++;

   }

   @Override
   public String getBirthday() {
      return birthday;
   }
}
```

```
//=============================================================================
=================================================================

//=============================================================================
=================================================================


class Official extends Recipient{
   protected String designation;


   public Official(String[] arr) {
      this.type = "Official";
      this.name = arr[0];
      this.email = arr[1];
      this.designation = arr[2];
   }
}


//=============================================================================
=================================================================

//=============================================================================
=================================================================


class Office_friend extends Official implements EmailReceiver{
   private String birthday;
   public Office_friend(String[] arr) {
      super(arr);
      this.type = "Office_friend";
      this.birthday = arr[3];


      number_of_recipients++;
```

```
    }

    @Override
    public String getBirthday() {
        return birthday;
    }
}

//============================================================================
=================================================================

//============================================================================
=================================================================

import java.io.Serializable;

class Email implements Serializable {
    //transient private static int count = 0;
    private static final long serialVersionUID = -2374314867706090224L;
    transient private final static String myAccount = "thilaknakumaratunga@gmail.com";
    private String to;
    private String date;
    private String subject;
    private String message;

    public String getDate() {
        return date;
    }
```

```
    //    public static void setCount() {
//        Email.count = count+1;
//    }

//    public static int getCount() {
//        return count;
//    }

    public Email(String to,String date, String subject, String message) {
        this.to = to;
        this.date = date;
        this.subject = subject;
        this.message = message;
    }

    public String getDetails(){
        String details = "Message Details:\n" +
                "Sender: "+myAccount+"\n" +
                "Recipient: "+to+"\n" +
                "Subject: "+subject+"\n" +
                "Message: "+message+"\n";
        return details;
    }

}

//=========================================================================================
=================================================================
```

```
//============================================================================
============================================================

import javax.mail.*;
      import javax.mail.internet.InternetAddress;
      import javax.mail.internet.MimeMessage;
      import javax.sound.midi.Soundbank;
      import java.io.IOException;
      import java.io.Serializable;
      import java.util.Properties;
      import java.time.LocalDate;


/**
 * this class does the all the tasks linked with sending emails(birthday greetings and custom emails)
 *     creating an email
 *     sending an email
 *     serialize the email by calling the relevant methods
 **/
class EmailOperations{


   private final static String myAccount = "thilaknakumaratunga@gmail.com";
   transient private final static String password = "mxztqdqfsshqvmig";


   /**
    * this method send the email to the recipient by connecting to the gmail server
    * **/
   public  static void send(String recipient,String subject, String text) throws MessagingException,
IOException {
```

```
Properties properties = new Properties();

properties.setProperty("mail.smtp.host","smtp.gmail.com");
// connect to the relevant port
properties.setProperty("mail.smtp.port","587");
// enabling a secure connection
properties.put("mail.smtp.starttls.enable", "true");
// enable authentication
properties.setProperty("mail.smtp.auth", "true");



Session session = Session.getInstance(properties, new Authenticator() {
   @Override
   protected PasswordAuthentication getPasswordAuthentication() {
      return new PasswordAuthentication(myAccount,password);
   }
});
Message message = createMessage(session, recipient,subject,text);
assert message != null;
try {
   Transport.send(message);
   System.out.println("Your message has been sent!");

   // Serialize the email
   String date = String.valueOf(LocalDate.now());
   Email email = new Email(recipient,date,subject,text);
   SerializingAndDeserializingOperations.serialize(email);

}catch (SendFailedException ex){
```

```
            System.out.println("Address not found!\n");
            return;
        }

    }

    /**
     * this method used to send birthday greeting on the current day
     */
    public static void sendGreetings(EmailReceiver receiver, String type, String name) throws
MessagingException, IOException {


        String message;
        String recipient;
        String subject = "Greeting";

        if(type.equals("Office_friend")){
            Office_friend officialFriend = (Office_friend) (receiver);
            message = "Wish you a happy birthday!!\n"+name;
            recipient = officialFriend.getEmail();
            EmailOperations.send(recipient,subject,message);
        }
        else if(type.equals("Personal")){
            Personal personal = (Personal) (receiver);
            message = "Hugs and love on your birthday!!\n"+name;
            recipient =  personal.getEmail();
            EmailOperations.send(recipient,subject,message);
        }
```

```
    }



    /**
     * this method creates a message and set the corresponding values in the message
     **/
    private static Message createMessage(Session session, String recipient, String subject, String text)
throws MessagingException, IOException {


        try {
            Message message = new MimeMessage(session);
            message.setFrom(new InternetAddress(EmailOperations.myAccount));
            message.setRecipient(Message.RecipientType.TO, new InternetAddress(recipient));
            message.setSubject(subject);
            message.setText(text);
            return message;
        } catch (Exception ex) {
            ex.printStackTrace();
            return null;
        }


    }


}
//=============================================================================
=============================================================
```

```
//============================================================================
============================================================

class RecipientCreator {

   public static Recipient createRecipient(String type,String[] data) {
      Recipient recipient = null;
      if (type.equals("Official")) {
         recipient = new Official(data);

      }
      else if (type.equals("Office_friend")) {
         recipient = new Office_friend(data);
      }
      else if (type.equals("Personal")) {
         recipient = new Personal(data);
      }

      return recipient;
   }

}

//============================================================================
============================================================
//============================================================================
============================================================

import java.io.*;
/**
```

```
 * this class handles all the serialization and
 * deserialization of the email objects
 **/
class SerializingAndDeserializingOperations{
   private static File serFile;

   private SerializingAndDeserializingOperations() {

   }

   public static File getInstance() {
      String path = System.getProperty("user.dir");
      if( serFile== null) {
         try {

            serFile = new File(path+"\\sentEmailList.txt");

            serFile.createNewFile();
            serFile.canWrite();
            serFile.canRead();

         }
         catch (Exception e) {
            e.printStackTrace();
         }
      }

      return serFile;
   }
```

```
  /**
   * this method serializes an email and save it in a text file
   **/
  public static void serialize(Email email) throws IOException {

     FileOutputStream fileOutputStream = new FileOutputStream(serFile,true);

     // serialize the objects and save without a header which is given by ObjectOutputStream
     if(serFile.length()==0){
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);
        objectOutputStream.writeObject(email);
        objectOutputStream.close();
     }
     else{
        CustomObjectOutputStream objectOutputStream = new
CustomObjectOutputStream(fileOutputStream);
        objectOutputStream.writeObject(email);
        objectOutputStream.close();
     }

     fileOutputStream.close();
     System.out.println("Serialization done!!");

  }

  /**
   * this method deserializes the objects which are saved in the hard disk
   * and print the details of the emails prior to a given date
```

```
 **/
  public static void deserializeAndPrint(File file, String date) throws IOException,
ClassNotFoundException {
    String list_4[] = date.split("/");
    String Y = list_4[0];
    String M = list_4[1];
    String D = list_4[2];

    FileInputStream fileInputStream = new FileInputStream(serFile);

    if(serFile.length()==0){
      return;
    }

    ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);



    while(true)
    {
      try {
        Object object = objectInputStream.readObject();
        Email email = (Email)(object);

        //System.out.println(email.getDate());
        String[] sent_date= email.getDate().split("-");
        if(sent_date[0].equals(Y) && sent_date[1].equals(M) && sent_date[2].equals(D)){
          System.out.println(email.getDetails());
        }
```

```
            }
            catch (EOFException ex){
               System.out.println("Finished!");
               break;
            }
            catch (Exception e){
               e.printStackTrace();
            }

         }

         objectInputStream.close();
         fileInputStream.close();
         //System.out.println("Deserialization done!!");

      }




}
/**
 * this is custom object output stream class which overrides the ObjectOutputStream
 * and this will not add a header part when serializing
 **/
class CustomObjectOutputStream extends ObjectOutputStream {

   CustomObjectOutputStream() throws IOException
```

```
  {
    super();
  }
  CustomObjectOutputStream(OutputStream outputstream) throws IOException
  {
    super(outputstream);
  }
  public void writeStreamHeader() throws IOException
  {
    return;
  }
}

//============================================================================
=================================================================
//============================================================================
=================================================================

import java.io.*;
    import java.util.ArrayList;

/**
 * This class does all the operations linked with the
 * recipient database text file
 */

class RecipientDataTextFileOperations {

  /**
```

```
 * Email client have a single text file to insert and retrieve data
 * **/
private static File fileObject;



private RecipientDataTextFileOperations() {


}


public static File getInstance() {
   String path = System.getProperty("user.dir");
   if( fileObject== null) {
      try {

         fileObject = new File(path+"\\clientList.txt");


         // if there is no such file create a new file
         fileObject.createNewFile();


         fileObject.canWrite();
         fileObject.canRead();


      }
      catch (Exception e) {
         e.printStackTrace();
      }
   }


   return fileObject;
```

```
}

/**
 * this method writes the user input(recipient details)
 * in to the text file
 **/
public static void write(String string) throws IOException {
   BufferedWriter writer = new BufferedWriter(new FileWriter(fileObject,true));
   writer.write(string+"\n");
   writer.close();
}

/**
 * this method loads the recipients in to the application
 * by creating objects based on their types
 **/
public static ArrayList<Recipient> load_recipients(ArrayList<Recipient> array) throws IOException {

   BufferedReader reader = new BufferedReader(new FileReader(fileObject));
   String details  =  null;
   try {
      while((details = reader.readLine())!=null){

         details.replace("\n","");
         String[] list_1 = details.split(":");



         //taking the type of the recipient
         String type = list_1[0].trim();
```

```
            //other details (name,email,etc)
            String[] data = list_1[1].split(",");

            //call the recipient creator and add the returned object to the array
            array.add(RecipientCreator.createRecipient(type,data));
          }
          reader.close();
       }catch (Exception e){

       }

       return array;
    }

    /**
     * this method is used to load the recipients whom a birthday greeting
     * should be sent, to the application
     **/
    public static ArrayList<EmailReceiver> load_greetings_receivers(ArrayList<EmailReceiver> array)
throws IOException {
       BufferedReader reader = new BufferedReader(new FileReader(fileObject));
       String details = null;
       while((details = reader.readLine())!=null){

          details.replace("\n","");

          String[] list_1 = details.split(":");
```

```
            //taking the type of the recipient
            String type = list_1[0].trim();


            //other details (name,email,etc)
            String[] data = list_1[1].split(",");


            // add the greeting receivers to the array
            if(type.equals("Office_friend") || type.equals("Personal"))
                array.add((EmailReceiver) RecipientCreator.createRecipient(type,data));
        }


        reader.close();
        return array;


    }


}
//==================================================================================================================================================================
//==================================================== END OF THE PROGRAM ============================================================================
//==================================================================================================================================================================
```