Class - Recipient.java

```
package OOP_Proj_1;  //190199A
//Providing base class for all recipients.
//We do not want to create objects, so we declare it as abstract

abstract class Recipient {

    private static int no_of_recipient;
     // To have the count of number of recipients
    private String name;
    private String mail_address;
// ------------------------------------------------------------------------------------------------------

    //constructor
    public Recipient(String inp_name, String inp_mail) {
       this.name = inp_name; //setting the name …
       this.mail_address = inp_mail; //setting the mail
       no_of_recipient = get_No_of_recipient() + 1;
       //when creating object, count should be increased
    }
// ------------------------------------------------------------------------------------------------------

       public String getName() { //getter for name
              return name;
       }

       public String getMail_address() { //getter for mail address
              return mail_address;
       }

       public static int get_No_of_recipient() { //getter for the count of number of recipients
              return no_of_recipient;
       }
       //no setters as this are only read only
}
```

Class - Official_recipient.java

```java
package OOP_Proj_1;

//Official recipient is a sub class of recipient

class Official_recipient extends Recipient {
	private String designation;

// ---------------------------------------------------------------------------------------------------

	//Constructor

    public Official_recipient(String inp_name, String inp_mail, String inp_designation) {

	super(inp_name,inp_mail);
	//Calling super class constructor to initialize name and mail
	this.designation = inp_designation;
    }

// ---------------------------------------------------------------------------------------------------
	public String getDesignation() {
	 //getter for designation
		return this.designation;
	}
	//no setters as this is only read only

}
```

Class - Official_close_friend.java

```java
package OOP_Proj_1;

import java.time.LocalDate; //since this class has date attrinute

//subclass of Official_recipient and also implements Birthday_wish_sendable interface..

class Official_close_friend extends Official_recipient  implements Birthday_wish_sendable{
	private LocalDate birth_date; //encapsulation
// ------------------------------------------------------------------------------------------------

	//constructor
    public Official_close_friend(String inp_name, String inp_mail, String inp_designation, LocalDate inp_birthday) {

	super(inp_name, inp_mail, inp_designation);
	//calling super class constructors initialize
	this.birth_date = inp_birthday;
    }
// ------------------------------------------------------------------------------------------------

  //no setters as this is  read only
	public LocalDate getBirth_date() { //getter for get birthday
		return birth_date;
	}

//implementing interface method ..
	@Override
	public String get_wish_text() {
	//function to send the body of the birthday wish email
		return "Wish you a Happy Birthday. With regards, Gopi";
	}
}
```

Class - Personal_recipient.java

```java
package OOP_Proj_1;

import java.time.LocalDate; //since this class has date attribute

//subclass of Recipient and also implements Birthday_wish_sendable interface..

class Personal_recipient extends Recipient implements Birthday_wish_sendable {
	private String nick_name; //encapsulation
	private LocalDate birth_date; //only getters.. no setters as read only
// ------------------------------------------------------------------------------------------------

    public Personal_recipient(String inp_name,String inp_nick_name, String inp_mail, LocalDate inp_birthday) {

	super(inp_name,inp_mail);
	this.nick_name = inp_nick_name;
	this.birth_date = inp_birthday;
    }

 // ------------------------------------------------------------------------------------------------

	public String getNick_name() { //getters for nickname
		return nick_name;
	}

	public LocalDate getBirth_date() { //getters for birthdate
		return birth_date;
	}
	//implementing interface method ..
	//function to send the body of the birthday wish email
	@Override
	public String get_wish_text() {
		return "hugs and love on your birthday. Lovely wishes from Gopi";
	}
}
```

Interface - Birthday_wish_sendable.java

```java
package OOP_Proj_1;

import java.time.LocalDate;

interface Birthday_wish_sendable {

	//To get body of the bithday wish text
	public String get_wish_text();

	//To get birthdate of recipients who can be wishable
	public LocalDate getBirth_date() ;
}
```

Email_Client.java

```java
// My index number : 190199A
// Shanmugavadivel Gopinath
// OOP Project – 1
// This is responsible to get user input
// This class consists main method

package OOP_Proj_1;
//import libraries
import java.io.IOException;// for error handling
import java.util.Scanner; //for getting inputs
import javax.mail.MessagingException;  //for handling inputs which can be occured during mail sending process

// ------------------------------------------------------------------------------------------------------


class Email_Client {

	public static void main(String[] args) throws IOException, MessagingException {
	  //creating an object of email process system to handle user's needs

		Email_processing_system mail_system = new Email_processing_system();
		//When programme start, it want to load the recipients form text file stored in hard disk.
		mail_system.load_recipients();


		////When programme start, it want to load the old mails form serilization file  stored in hard disk.
		mail_system.load_mails();


		//When programme start, it want to send mails to people who have birthday on that day
		mail_system.wish_today_birthday();

// ------------------------------------------------------------------------------------------------------
```

```
            Scanner scanner = new Scanner(System.in); //Creating Scanner objects to receive inputs
            Scanner scanner1 = new Scanner(System.in);


            System.out.println("Enter option type: \n" //to ask user to select the functionality
                  + "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(); //getting input to ask user to select the functionality

// ------------------------------------------------------------------------------------------------------
            switch(option){


                  case 1: //if we want to add a new recipient
                     //Input format is same as in text file
                     System.out.println("Enter Your new entry in classical format: ");
                     String new_recipient_details = scanner1.nextLine();
                         //getting inputs
                    // asking mail system to handle inputs and create new recipient
                     mail_system.create_recipient_entry(new_recipient_details);
                      break;

// ------------------------------------------------------------------------------------------------------

                  case 2: //If you want to Send an email
                     System.out.println("Enter Your new email details... input format - email, subject, content ");
                     String inp_for_mail_send = scanner1.nextLine(); //gettign inputs
                     //asking mail system to handle send request
                     mail_system.send_mail(inp_for_mail_send);
                      break;
```

```
// ------------------------------------------------------------------------------------------------------------


                case 3: // If you want to Print out all the recipients who have birthdays
                    System.out.println("Enter the date in yyyy/MM/dd form: ");
                    String input_date = scanner1.nextLine(); //getting the user input
                   //asking mail system to print out
                    mail_system.check_birthday(input_date);
                    break;

// ------------------------------------------------------------------------------------------------------------

                case 4: //Printing out details of all the emails sent on particular date
                    System.out.println("Enter the date in yyyy/MM/dd form: ");
                    String input_date1 = scanner1.nextLine(); //getting the date
                    //asking mail system to print out the query
                    mail_system.check_email_this_date(input_date1);
                    break;

 // ------------------------------------------------------------------------------------------------------------

                case 5: // Printing out the number of recipient objects in the application
                    System.out.println("Total Number of recipient object in the application is.... finding......");
                    //asking mail system to print out the query
                    System.out.println(mail_system.find_total_recipients());
                    break;


            }
            scanner.close();            //closing the scanners
            scanner1.close();


        }
}
```

Class- Email_processing_system.java

```java
//This object of this class recieves user queries from Email_cilent class .. and does the processing
// When the process is smaller, Object is sufficient to process the request
// If the process is larger it calls sub_division processing objects to do processing

//Importing the packages
package OOP_Proj_1;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;  // Import the IOException class to handle errors
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;
import javax.mail.MessagingException;

class Email_processing_system {
	//Attributes
	ArrayList<Birthday_wish_sendable> birthday_wishers; // Polymorphism - We store Birthday_wish_sendable objects
here
	Mail_history_processor mail_history_processor; // this processes the mail history and serialize mails
	Recipient_entry_adder recipient_entry_adder; // This creates new reciepients

// ------------------------------------------------------------------------------------------------

	//When we create email_processing_system object, We want other helping objects also want to be initiated
	public Email_processing_system() {
		mail_history_processor = new Mail_history_processor();
		recipient_entry_adder = new Recipient_entry_adder();
		birthday_wishers = new ArrayList<Birthday_wish_sendable>();
	}


// ------------------------------------------------------------------------------------------------
```

```java
	//When programme starts, it should load recipients from text file...
	//We need to read information from text file and call the sub division (recipient_entry_adder)
	// recipient_entry_adder will process the infromation and create recipients
	public void load_recipients() throws IOException {
		try {
			File myObj = new File("src/clientList.txt"); //file path
			Scanner myReader = new Scanner(myObj); //reading text file

			while (myReader.hasNextLine()) {
			  String data = myReader.nextLine();
			  //calling recipient_entry_adder.. this will process and return the recipient object

			  Recipient recipient1  = recipient_entry_adder.recipient_object_creator(data,true);
			  //if the recipient object can be wishable, then add it to our array list
				if(recipient1 instanceof Birthday_wish_sendable) {
					birthday_wishers.add((Birthday_wish_sendable)recipient1);
			  }

			}
			  myReader.close(); //closing the file writer
		}
		catch (FileNotFoundException e) { //Handiling exception
			System.out.println("An error occurred.");
			e.printStackTrace();
		}

	}



// ------------------------------------------------------------------------------------------------------

	//When programme starts, it should send wish email to people who have birthday today
	public void wish_today_birthday() throws MessagingException {
		LocalDate today_date = LocalDate.now(); //getting today date

		for(Birthday_wish_sendable obj : birthday_wishers ) {	//itearting birthday_wishers arraylist
			LocalDate temp_date = obj.getBirth_date(); //get birthdate of object from abstract function
```

```
                    //if today and object's month& date values are equal
                    if(temp_date.getMonthValue()==today_date.getMonthValue() &
temp_date.getDayOfMonth()==today_date.getDayOfMonth()) {
                          String body = obj.get_wish_text(); //get birthday-mail body from method of interface
                          String subject = "Happy Birthday!!!!"; //our subject
                         //  we need to cast from the interface reference to recipients reference to get address...
                          Recipient casted_obj = (Recipient)obj;
                          String reciever_mail = casted_obj.getMail_address();  // we get mail address using getter
                          Mail bd_mail = new Mail(reciever_mail,subject,body );  // we create new mail object
                          bd_mail.sendMail(); //we send the mail
                          mail_history_processor.add_to_history(bd_mail); //we add the sent mail to history
                    }
              }
      }

// ------------------------------------------------------------------------------------------------------

      //When programme starts, it should load mail objects from serialization file...
      //we should call the sub division (mail_history_processor)
      //That will serialize mail objects from mail
      public void load_mails() {
            mail_history_processor.load_old_mails();
      }

// ------------------------------------------------------------------------------------------------------

      //If user selects one ...
      //if we want to add a new recipient , we should call  sub division (recipient_entry_adder)
      // recipient_entry_adder will process the information and create recipients
      public void create_recipient_entry(String new_recipient_details) throws IOException {
            Recipient recipient  = recipient_entry_adder.recipient_object_creator(new_recipient_details,false);
            if(recipient instanceof Birthday_wish_sendable) {
                  birthday_wishers.add((Birthday_wish_sendable)recipient);
            };
}

// ------------------------------------------------------------------------------------------------------
      //If user selects two... Send Email..
      //We trim and split the user input and we create new mail object and send it
```

```
	public void send_mail(String inp_for_mail_send) throws MessagingException {
		String[] mail_data = inp_for_mail_send.split(",");  //We extract information
		if(mail_data.length==3) {//checking whether valid input
			String reciever_mail = mail_data[0].trim(); // mail address
			String subject = mail_data[1].trim(); //subject
			String body = mail_data[2].trim(); //body
			Mail new_mail = new Mail(reciever_mail,subject,body ); // creating new mail object
			new_mail.sendMail(); //we send the mail using method of it own
			mail_history_processor.add_to_history(new_mail);
			//After sending, we need to add to the history,,so we call mail_history_processor
		}
	}



// ------------------------------------------------------------------------------------------------------


	//If user selects three... Print out all the recipients who have birthdays in particualar day
	public void check_birthday(String input_date) {
		DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy/MM/dd"); //Defining date format
		LocalDate recipient_birthdate = LocalDate.parse(input_date,formatter); //Changing string to date object
		System.out.println("Names of recipients who have their birthday set to current date"); //output
		 for(Birthday_wish_sendable obj : birthday_wishers ) { //iterate thorough our birthday_wishers arraylist

			LocalDate temp_date = obj.getBirth_date(); //getBirth_date() abstract function to get birthdate
			if(temp_date.getMonthValue()==recipient_birthdate.getMonthValue() &
temp_date.getDayOfMonth()==recipient_birthdate.getDayOfMonth()) {


				//If month& date values are equal  then we need to cast from the interface reference to
				recipients reference...
				 Recipient casted_obj = (Recipient)obj;
				//By casting we can access the names of the recipients
				 System.out.println(casted_obj.getName());  // calling getters to print names
			}
		 }
	}
```

```
// ---------------------------------------------------------------------------------------------------

	//If user selects four... Printing out details of all the emails sent on particular date
	public void check_email_this_date(String input_date) {
		DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy/MM/dd"); //We define the pattern
		LocalDate date = LocalDate.parse(input_date,formatter); //convert string to date object
		mail_history_processor.check_mails_this_day(date); // call mail_history_processor to print the mail info

	}

// ---------------------------------------------------------------------------------------------------

	/// If user selects five, Printing out the number of recipient objects in the application
	// we keep a static variable in recipient base class to keep count
	public int find_total_recipients() {
		return Recipient.get_No_of_recipient(); //we call the getter
	}

}
```

Class - Mail.java

```java
package OOP_Proj_1;
// THis class is responsible for creating mail objects
// This class describes the functionality of mail obejcts eg: mail send
//importing libraries needed for sending mail
import java.io.Serializable;
import java.time.LocalDateTime;
import java.util.Properties;
import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

class Mail  implements Serializable {
	//because mail object needed to be serializable
	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	private String recepient; //Attributes of a Mail object
	private String subject;
	private String text; //body of mail
	private LocalDateTime sent_time; //time when mail has sent

// ----------------------------------------------------------------------------------------------------
	//Constructor
	public Mail(String inp_recepient,  String inp_subject, String inp_text) {
		//initializing mail objects
		this.recepient = inp_recepient;
		this.subject = inp_subject;
		this.text = inp_text;
```

```
	}

	//Here we use getters... We need to set only time.. So we define setter only for time
	public String getRecepient() {
		return recepient;
	}

	public String getSubject() {
		return subject;
	}

	public String getText() {
		return text;
	}

	public LocalDateTime getSent_time() {
		return sent_time;
	}

// ------------------------------------------------------------------------------------------------------

//THis method describes the sendMail function of mail objects
	public void sendMail() throws MessagingException{

		System.out.println("Message is prepareing to send..."); //SIgnaling

		Properties properties = new Properties(); //describing properties for gmail

		properties.put("mail.smtp.auth", "true");
		properties.put("mail.smtp.starttls.enable", "true");
		properties.put("mail.smtp.host", "smtp.gmail.com");
		properties.put("mail.smtp.ssl.trust", "smtp.gmail.com");
		properties.put("mail.smtp.port", "587");

		//cilent's mail information
		String mailAddress = "shangopimora@gmail.com";
		String password = "gopinath";
```

```
            //Creating Session object
            Session session = Session.getInstance(properties,new Authenticator() {
                  @Override
                  protected PasswordAuthentication getPasswordAuthentication() {
                        return new PasswordAuthentication(mailAddress, password);
                  }
            });

            //Sending the messege to next function
            Message msg = prepareMessage(session, mailAddress, this.recepient, this.subject, this.text);
            this.sent_time = LocalDateTime.now();  //Assigning time for the mail objects

            Transport.send(msg); //send

            System.out.println("Message sent successfully..."); //successfully sent
      }

// ------------------------------------------------------------------------------------------------------
      //THis static function sends the messege provided
      private static Message prepareMessage(Session session, String mailAddress, String recepient, String subject,
String Text) {
            Message msg = new MimeMessage(session);
            try {
                  msg.setFrom(new InternetAddress(mailAddress));
                  msg.setRecipient(Message.RecipientType.TO, new InternetAddress(recepient));
                  msg.setSubject(subject); //sets subject
                  msg.setText(Text); //sets Text
                  return msg;

            } catch (AddressException e) { //if mail address is wrong
                  e.printStackTrace();
            } catch (MessagingException e) {
                  e.printStackTrace();
            }
            return null;
      }


}
```

Class – Sub_division.java

```java
package OOP_Proj_1;

abstract class Sub_division {
	//We have a Email processing Class as backend process
	//It implements the functionalities  ..
	// When the implementation is bigger, Solving all process in same process(Class) is not good
	// So Big implementation are moved to small divisions
	// All small divisions run under this class
	// We have Mail_history_Processor, Recipient_entry_adder subdivisions are under this class

}
```

Class - Recipient_entry_adder.java

```
//This class is a subclass of sub_division is responsible for processing user inputs and create new reciepient
objects
//When program start, it needed to load all recipient objects from text file.. This class has methods to do it,,

package OOP_Proj_1;
//importing input output libraries and Date libraries
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;


class Recipient_entry_adder extends Sub_division {
		//This class don't have any Attributes
// ------------------------------------------------------------------------------------------------------

	// We need to process user input and create & return recipient objects
	public Recipient recipient_object_creator(String new_recipient_details, boolean is_loading) throws IOException{
		// new_recipient_details Inputs will be in this format
		// Official: nimal,nimal@gmail.com,ceo
		// Office_friend: kamal,kamal@gmail.com,clerk,2000/12/12
		//Personal: sunil,<nick-name>,sunil@gmail.com,2000/10/10

		//We will call this function in two incidents...
		//When programme starts, we need to load recipients from text files.. in that time is_loading = true
		//When user wants to create new recipient, in that time is_loading = false
// ------------------------------------------------------------------------------------------------------
		Recipient recipient = null; //initalizing recipients



		//First We need to know what type of recipient.. So we split the String and assign it to an array
		String[] temp_array = new_recipient_details.split(" ");
```

```
            //temp_array[1] hold infomrmation regarding recipients... We split it to extract that information.
            String[] recipient_data = temp_array[1].split(",");
            int temp_array_size = temp_array.length; // It should be 2.. or we should raise error
            int recipient_data_size = recipient_data.length; // It should be constant for particular type of
recipients
            String test_case = temp_array[0].toLowerCase(); //We should neglect case difference
// ------------------------------------------------------------------------------------------------------
            //Here we want to write the new recipient to text files too
            File file = new File("src/clientList.txt"); // Path if text file
            FileWriter writer_1 = new FileWriter(file, true); //We want to append ...so we assign true


// ------------------------------------------------------------------------------------------------------
            //If official
            if(test_case.startsWith("official") & temp_array_size==2 & recipient_data_size==3) {
                  String recipient_name= recipient_data[0]; //name
                  String recipient_mail = recipient_data[1]; //mail
                  String recipient_designation = recipient_data[2]; //designation
                  Official_recipient official_recepient = new Official_recipient(recipient_name,
recipient_mail,recipient_designation);
                  recipient = official_recepient;                     //We create official recipient object

                  if(!is_loading) {  //if we are creating objects from user input and not from text file
                  writer_1.write(new_recipient_details); // We want to write it to text file
                  System.out.println("Official recipient Entry added successfully!!! Your last entry was " +
official_recepient.getName());
                  }
                        }

// ------------------------------------------------------------------------------------------------------


            //If office_friend
            else if(test_case.startsWith("office_friend") & temp_array_size==2 & recipient_data_size==4) {
                  String recipient_name= recipient_data[0]; //name
                  String recipient_mail = recipient_data[1]; //mail
                  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy/MM/dd"); //Date formatter style
                  String recipient_designation = recipient_data[2]; //designation
                  LocalDate recipient_birthdate = LocalDate.parse(recipient_data[3],formatter);  //Formatting bd
```

```
                    Official_close_friend official_friend = new Official_close_friend(recipient_name, recipient_mail,
recipient_designation, recipient_birthdate);
                    recipient=official_friend; //Creating Official friend object

                    if(!is_loading) {//if we are creating objects from user input and not from text file
                          writer_1.write(new_recipient_details); // We want to write it to text file
                          System.out.println("Official friend recipient Entry added successfully!!! Your last entry
was " + official_friend.getName());
                          }

             }
// ------------------------------------------------------------------------------------------------

             //If personal reciepeint
             else if(test_case.startsWith("personal")& temp_array_size==2 & recipient_data_size==4 ) {
                    String recipient_name= recipient_data[0]; //name
                    String recipient_nickname = recipient_data[1]; //nick -name
                    String recipient_mail = recipient_data[2]; //mail
                    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy/MM/dd"); //Date formatter style
                    LocalDate recipient_birthdate = LocalDate.parse(recipient_data[3],formatter);   //Formatting bd
                    Personal_recipient personal_recipient = new Personal_recipient(recipient_name, recipient_nickname,
recipient_mail, recipient_birthdate);
                    recipient = personal_recipient; //Creating personal friend object

                    if(!is_loading) {  //if we are creating objects from user input and not from text file
                          writer_1.write("\n"+new_recipient_details ); // We want to write it to text file
                          System.out.println("Personal friend recipient Entry added successfully!!! Your last entry
was " + personal_recipient.getName());
                          }

             }
             else { // If don't include in any of above cases, Then there should be problem with the input
                    System.out.print("Invalid Input. Please Try again later");
             }

             writer_1.close(); //CLosing the writer
             return recipient; //We return the reciepient object created now
      }
}
```

Class – Mail_history_processor.java

```java
//This class is a subclass of sub_division is responsible for managing history of sent mail
//This will serialize and deserialize Mail object in ArrayList structure between JVM and Local machine

package OOP_Proj_1;
//importing libraries needed for serialization
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.time.LocalDate;
import java.util.ArrayList;

class Mail_history_processor extends Sub_division   {
	ArrayList<Mail> mail_history; // THis arraylist consists the history of sent mail
// ------------------------------------------------------------------------------------------------------

	//When Sub divison get activated it should create new ArrayList.. We create in constrctor
	public Mail_history_processor() {
		mail_history = new ArrayList<Mail>();
	}
// ------------------------------------------------------------------------------------------------------
		//When user sends email, we need to store to history
	//THis method is used to add mails to the history
	public void add_to_history(Mail mail) {
		mail_history.add(mail); //We add this in our arraylist
		//We have to write it in local file (Serialize)
		try {
		  FileOutputStream fop=new FileOutputStream("object.ser"); //file path
		  ObjectOutputStream oos=new ObjectOutputStream(fop);
		  oos.writeObject(mail_history); //here we serialize Mail object in ArrayList form
		  oos.close(); //closing outputstream
		} catch (IOException e) { //Catch some exception
		  System.out.println(e);
	}
```

```
	}

// ------------------------------------------------------------------------------------------------

	//When programme start, we need to read the serialzied file and want to create mail objects
	//We want to de-serialize the object.ser file
	public void load_old_mails(){
		try {
			FileInputStream fis=new FileInputStream("object.ser"); //file path
			ObjectInputStream ois=new ObjectInputStream(fis);
			mail_history =(ArrayList<Mail>)ois.readObject(); //here we read the file and cast it as
Arraylist<mail>
			//We have now de-serialized mail objects from file in the form of ArrayList Structure
			ois.close(); //We close input input stream
	}

		catch(Exception e) {//Catch some exception
			System.out.println("Error");
		}
	}


// ------------------------------------------------------------------------------------------------

	//Sometimes mail_processing_system needs information of mails sent on particular date ..Here we process it
	public void check_mails_this_day(LocalDate date) {
		for(Mail mail : mail_history) { //We itearte our mail_history ArrayList
			if(mail.getSent_time().toLocalDate().equals(date)) { //If given date and mail_sent_date are equal
				System.out.println("Subject : "+ mail.getSubject()); //Then give mail information
				System.out.println("Recipient : "+ mail.getRecepient());
			}

		}

	}
}
```