```java
package net.codejava;

import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Base64;
import java.util.Date;
import java.util.Iterator;
import java.util.List;
import java.util.Locale;
import java.util.Scanner;
import java.util.stream.Collectors;

import javax.activation.MimetypesFileTypeMap;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;

import org.apache.commons.io.FileUtils;
import org.json.JSONArray;
import org.json.JSONObject;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.DeserializationFeature;
import com.fasterxml.jackson.databind.JavaType;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.mailjet.client.MailjetClient;
import com.mailjet.client.MailjetRequest;
import com.mailjet.client.MailjetResponse;
import com.mailjet.client.errors.MailjetException;
import com.mailjet.client.errors.MailjetSocketTimeoutException;
import com.mailjet.client.resource.Contact;
import com.mailjet.client.resource.Contactdata;
import com.mailjet.client.resource.Message;
import com.mailjet.client.resource.Template;
import com.opencsv.CSVReader;
import com.opencsv.CSVReaderBuilder;
import com.opencsv.exceptions.CsvException;

import net.codejava.model.ContactDetail;
import net.codejava.model.ContactMessage;
import net.codejava.model.ContactModel;
import net.codejava.model.EmailTemplate;

public class Main {

    private static final String        clePublic = "eb012fe8dfda1d693cb402ec1f28573c";
    private static final String        clePrive  = "abc30595f0f61101b0887cc726e220f1";
    private static final MailjetRequest requestGetMail     = new MailjetRequest(
    Message.resource).filter("limit", 0);
    private static final MailjetRequest requestGetTemplate = new MailjetRequest(
    Template.resource).filter("limit", 0);
    private static final MailjetRequest requestGetData     = new MailjetRequest(
    Contactdata.resource).filter("limit", 0);
    private static final MailjetRequest requestGetContact  = new MailjetRequest(
    Contact.resource).filter("limit", 0);
    private static final MailjetClient client    = new MailjetClient(clePublic,
    clePrive);

    public static void main(String[] args) throws MailjetException,
    MailjetSocketTimeoutException, IOException, CsvException {
        String choise;
        Boolean continuer = true;
        List<String[]> contactListCsv = null;

        try (Scanner myObj = new Scanner(System.in)) {

            while (continuer) {
```

```
64                  System.out.println("Enter 0 to see the menu\r\n");
65                  choise = myObj.nextLine();
66                  if (choise.equals("0")) menu();
67                  if (choise.equals("1")) emailControler(myObj, attachedFileOrNot(myObj
                    ), contactListCsv);
68                  if (choise.equals("2")) getMails();
69                  if (choise.equals("3")) getMailSendToOneUser(myObj);
70                  if (choise.equals("4")) menuNewContact(myObj);
71                  if (choise.equals("5")) seeContact();
72                  if (choise.equals("6")) csvImpoter(myObj);
73                  if (choise == "") continuer = false;
74              }
75          } catch (NumberFormatException e) {
76              e.printStackTrace();
77          }
78          System.out.println("Bye");
79      }
80
81      public static void menu() {
82          System.out.println("This is the order you can use" + "\r\n1-Write Email" +
            "\r\n2-Mail you send today"
83                  + "\r\n3-Have the Email you send to a contact" + "\r\n4-Make a new
                    contact" + "\r\n5-See your contacts"
84                  + "\r\n6-Importe a contact list (.csv)\r\n" + "\r\npress enter to
                    quit\r\n");
85      }
86
87      // ---------To see all contact and receive the list of contact made by
88      // seeContact
89      public static List<ContactModel> menuMail() throws MailjetException,
        MailjetSocketTimeoutException {
90          List<ContactModel> contactList;
91
92          System.out.println("You can send mail to its customers (this is all you
            contact) :\n");
93          contactList = seeContact();
94
95          System.out.println("Enter the name \nFor stop press enter\n");
96          System.out.println("\nFor all contact tap all\n");
97          return contactList;
98      }
99
100     // ---------To create a new contact, that can be stop, the contact need a
101     // name and a first name
102     // ---------This function check if the mail use already and if not he create
103     // a contact with the mail and add the name and the firstName to the
104     // properties
105
106     public static void menuNewContact(Scanner myObj) throws MailjetException,
        MailjetSocketTimeoutException {
107         String mailExist = null;
108         Boolean itsOK = false;
109         String nomContact ;
110         String prenom = "";
111         String mailContact;
112         while (!itsOK) {
113             prenom = firstName(myObj);
114             nomContact = name(myObj);
115             System.out.println("Enter the mail of your contact");
116             mailContact = myObj.nextLine();
117             if (isValidEmailAddress(mailContact)) {
118                 mailExist = searchAContact(mailContact.toLowerCase().replace(" ", ""
                    ));
119                 System.out.println(mailExist);
120                 if (mailExist.length() == 0) {
121                     createNewContact(mailContact, nomContact);
122                     final int contactId = Integer.parseInt(searchAContact(mailContact.
                        toLowerCase().replace(" ", "")));
123                     addProperties(contactId, nomContact, "prenom");
```

```
124                     addProperties(contactId, prenom, "nom");
125                     itsOK = true;
126                 } else
127                     System.out.println("The contact you want to create already exist"
                        );
128             } else {
129                 System.out.println("The email you want to use is invalide");
130             }
131         }
132     }
133 
134     public static String firstName(Scanner myObj) {
135         String prenom = "";
136 
137         while (prenom == "") {
138             System.out.println("Enter the nickname of your contact");
139             prenom = myObj.nextLine();
140             if (prenom == "") System.out.println("invalide nickname");
141         }
142         return prenom;
143     }
144 
145     public static String name(Scanner myObj) {
146         String nomContact = "";
147         while (nomContact == "") {
148             System.out.println("Enter the contact name");
149             nomContact = myObj.nextLine();
150             if (nomContact == "") System.out.println("invalide nomContact");
151         }
152         return nomContact;
153     }
154 
155     // -------When in the menu you press one you chose to send a e-mail, if you
156     // tap all, that send to any contact you made
157     // -------or you can enter one by one the mail you want to use
158 
159     public static void manualMail(Scanner myObj, String oneMail, Boolean emailValidate
        , List<ContactModel> contactList, String subject,
160              String libTemplate, String[] AttachedFileOrNot) {
161         Boolean continueToAddUser = true;
162         oneMail = myObj.nextLine().toString();
163         if (oneMail.equals("all")) for (ContactModel cM : contactList) SendMail.
            sendMail(client, subject, libTemplate, true, cM.getEmail().toString(),
            AttachedFileOrNot);
164         while (continueToAddUser) {
165             if (!(oneMail.trim().isEmpty())) {
166                 for (ContactModel r : contactList) {
167                     emailValidate = r.getEmail().toString().equals(oneMail) ? true :
                        false;
168                     if (emailValidate) break;
169                 }
170                 if (emailValidate == true) {
171                     SendMail.sendMail(client, subject, libTemplate, true, oneMail,
                        AttachedFileOrNot);
172                 } else
173                     System.out.println("Email invalide");
174                 oneMail = myObj.nextLine().toString();
175             } else
176                 continueToAddUser = false;
177         }
178     }
179 
180     // when you tap 6 in the menu the import a .csv with your contact
181     // if the mail doesn't exist in the contact base, we create it
182     // if the mail is false,
183     public static void csvImpoter(Scanner myObj) throws IOException, CsvException,
        MailjetException, MailjetSocketTimeoutException {
184         String fileName = "C:\\Users\\DEV3\\Desktop\\dossier csv\\Contact.csv";
185         Integer i = 1;
```

```
186             String mailExist;
187             try (CSVReader reader = new CSVReaderBuilder(new FileReader(fileName)).
                withSkipLines(1).build()) {
188                 List<String[]> r = reader.readAll();
189                 for (String[] x : r) {
190                     if (!(x[0].equals("")) && isValidEmailAddress(x[0]) == true) {
191                         mailExist = searchAContact(x[0].toLowerCase().replace(" ", ""));
192                         if (mailExist.length() == 0) {
193                             System.out.println("The contact " + x[0] + " doesn't exist");
194                             createNewContactFromCSV(x[2], x[1], x[0], myObj);
195                         }
196                     } else if (x[0].equals(""))
197                         System.out.println("Email (n°" + i + ") non-existent\n");
198                     else
199                         System.out.println("Email " + x[0] + " invalide (n°" + i + ")\n");
200                     i++;
201                 }
202                 for (String[] x : r) {
203                     if (isValidEmailAddress(x[0])) System.out.println(x[0]);
204                 }
205                 System.out.println("Did you want to send a mail to this list ?(press
                    enter to say yes)");
206                 if (myObj.nextLine().equals("")) emailControler(myObj, attachedFileOrNot(
                    myObj), r);
207             }
208         }
209
210         // with that we can make a contact from the csv
211         public static void createNewContactFromCSV(String prenom, String nom, String
            mailDoesntExist, Scanner myObj)
212                 throws MailjetException, MailjetSocketTimeoutException {
213             if (prenom.length() == 0) {
214                 System.out.println("The conctact " + mailDoesntExist + " doesn't have a
                    first name, give him one");
215                 prenom = myObj.nextLine();
216             }
217             if (nom.length() == 0) {
218                 System.out.println("The conctact " + mailDoesntExist + " doesn't have a
                    name, give him one");
219                 nom = myObj.nextLine();
220             }
221             createNewContact(mailDoesntExist, nom);
222             addProperties(Integer.parseInt(mailDoesntExist), prenom, "prenom");
223             addProperties(Integer.parseInt(mailDoesntExist), nom, "nom");
224         }
225
226         // enter information for the two methode of sending mails (csv or
227         // "traditinal")
228         public static void emailControler(Scanner myObj, String[] AttachedFileOrNot, List<
            String[]> contactListCsv)
229                 throws MailjetException, MailjetSocketTimeoutException, IOException {
230             List<ContactModel> contactList = null;
231             String oneMail = null;
232             Boolean emailValidate = false;
233             System.out.println("Enter the subject of your mail");
234             String subject = myObj.nextLine();
235             String libTemplate = useTemplate(myObj);
236             if (contactListCsv == null) {
237                 contactList = menuMail();
238                 manualMail(myObj, oneMail, emailValidate, contactList, subject,
                    libTemplate, AttachedFileOrNot);
239             } else
240                 listEmail(subject, libTemplate, AttachedFileOrNot, contactListCsv);
241         }
242
243         // send a mail to one contact with attached file or not
244         public static void listEmail(String subject, String libTemplate, String[]
            AttachedFileOrNot, List<String[]> contactListCsv) {
245              for (String[] x : contactListCsv)
```

```
246             SendMail.sendMail(client, subject, libTemplate, true, x[0],
                AttachedFileOrNot);
247     }
248
249     @SuppressWarnings("unchecked")
250     public static String useTemplate(Scanner myObj)
251             throws MailjetException, MailjetSocketTimeoutException,
                JsonMappingException, JsonProcessingException {
252         Boolean ItsOK = false;
253         String idChoose = null;
254         MailjetResponse response = client.get(requestGetTemplate);
255         final List<EmailTemplate> messagesItems = (List<EmailTemplate>)
            getObjectListFromObject(response.getData().toString(), EmailTemplate.class);
256         messagesItems.forEach(x -> System.out.println(x.getiD() + " " + x.getName()));
257         while (!ItsOK) {
258             System.out.println("This your available template \nlink to create them:
                https://app.mailjet.com/templates/marketing\n");
259             messagesItems.forEach(x -> System.out.println(x.getiD() + " " + x.getName
                ()));
260             System.out.println("which one do you want to take (enter the number)");
261             idChoose = myObj.nextLine();
262             for (EmailTemplate e : messagesItems)
263                 ItsOK = e.getiD().equals(idChoose) ? true : false;
264             if (!ItsOK) System.out.println("Invalide number\r\n");
265         }
266         return idChoose;
267     }
268
269     public static String[] attachedFileOrNot(Scanner myObj) throws IOException {
270         System.out.println("Did you want to use attached file ?(yes for send attached
            file )");
271         if (myObj.nextLine().equals("yes")) return encoderForAttachedFile(myObj);
272         return null;
273     }
274
275     public static String[] encoderForAttachedFile(Scanner myObj) throws IOException {
276
277         System.out.println("Enter the name of your folder(ex : test.txt) :
            \r\nAvailable extension : .txt .jar .pdf .jpeg .png .gif .mp3 .mp4");
278         String titleFile = myObj.nextLine();
279
280         System.out.println("Enter the location of your file, start at the \r\nroot
            directory  (ex: C:\\Users\\DEV3\\Desktop)");
281         String path = myObj.nextLine() + "\\" + titleFile;
282
283         MimetypesFileTypeMap mimeTypesMap = new MimetypesFileTypeMap();
284
285         File file = new File(path);
286         String typeFile = mimeTypesMap.getContentType(file);
287
288         byte[] fileContent = FileUtils.readFileToByteArray(file);
289         String encodedString = Base64.getEncoder().encodeToString(fileContent);
290
291         return new String[] { titleFile, typeFile, encodedString };
292     }
293
294     // to see them mails we send today
295     @SuppressWarnings("unchecked")
296     public static void getMails() throws MailjetException,
        MailjetSocketTimeoutException, JsonMappingException, JsonProcessingException {
297         String aujourdhui = convertDate(new Date());
298         MailjetResponse response = client.get(requestGetMail);
299         final List<ContactMessage> messagesItems = (List<ContactMessage>)
            getObjectListFromObject(response.getData().toString(),
300                 ContactMessage.class);
301         List<ContactMessage> result = messagesItems.stream().filter(x -> convertDate(x
            .getArrivedAt()).equals(aujourdhui))
302                 .collect(Collectors.toList());
303         result.forEach(x -> {
```

```
304                 try {
305                     System.out.println(x.getArrivedAt() + " " + x.getStatus() + " " +
                        searchAContactWithId(x.getContactID()));
306                 } catch (MailjetException e) {
307                     e.printStackTrace();
308                 } catch (MailjetSocketTimeoutException e) {
309                     e.printStackTrace();
310                 }
311             });
312             System.out.println("hours is UTC+0, FRANCE is UTC+2");
313         }
314 
315         // get all the mail we send to one user, first we check if the contact email
316         // is valid
317         public static void getMailSendToOneUser(Scanner myObj)
318                 throws MailjetException, MailjetSocketTimeoutException,
                    JsonMappingException, JsonProcessingException, NumberFormatException {
319             System.out.println("Enter the mail you research");
320             String mailExist = null;
321             Boolean itsOK = false;
322             while (!itsOK) {
323                 mailExist = searchAContact(myObj.nextLine().toLowerCase().replace(" ", ""
                    ));
324                 if (!(mailExist.length() == 0))
325                     itsOK = true;
326                 else
327                     System.out.println("The contact you search doesn't exist");
328             }
329             getMailByIdContact(Integer.parseInt(mailExist));
330         }
331 
332         // see mail we send to a user
333         @SuppressWarnings("unchecked")
334         public static void getMailByIdContact(Integer id)
335                 throws MailjetException, MailjetSocketTimeoutException,
                    JsonMappingException, JsonProcessingException {
336             MailjetResponse response = client.get(requestGetMail);
337             final List<ContactMessage> messagesItems = (List<ContactMessage>)
                    getObjectListFromObject(response.getData().toString(),
338                     ContactMessage.class);
339             List<ContactMessage> result = messagesItems.stream().filter(x -> x.
                    getContactID().equals(id)).collect(Collectors.toList());
340             result.forEach(x -> System.out.println(x.getStatus() + " " + x.getArrivedAt
                    ()));
341         }
342 
343         // see all contact we get in the data contact base
344         @SuppressWarnings("unchecked")
345         public static List<ContactModel> seeContact() throws MailjetException,
            MailjetSocketTimeoutException {
346             MailjetResponse response1 = client.get(requestGetContact);
347             MailjetResponse response2 = client.get(requestGetData);
348             try {
349                 List<ContactModel> contactItems = (List<ContactModel>)
                    getObjectListFromObject(response1.getData().toString(), ContactModel.class
                    );
350                 final List<ContactDetail> contactDetailItems = (List<ContactDetail>)
                    getObjectListFromObject(response2.getData().toString(),
351                             ContactDetail.class);
352                 for (Iterator<ContactModel> iterator = contactItems.iterator(); iterator.
                    hasNext();) {
353                     ContactModel contact = (ContactModel) iterator.next();
354                     final ContactDetail currentDetail = contactDetailItems.stream().filter
                    (detail -> detail.getContactID().equals(contact.getId()))
355                             .findFirst().orElse(null);
356                     contact.setName(currentDetail.getDataValue("nom").getValue());
357                     contact.setFirstName(currentDetail.getDataValue("prenom").getValue());
358                     if (contact.isValid()) System.out.println(contact.getName() + " " +
                    contact.getFirstName() + " : " + contact.getEmail());
```

```
359                 }
360                 return contactItems.stream().filter(x -> !x.getEmail().startsWith("\\x")).
                    collect(Collectors.toList());
361             } catch (Exception e) {
362                 e.printStackTrace();
363             }
364             return null;
365         }
366 
367         public static void createNewContact(String mailContact, String nomContact) throws
        MailjetException, MailjetSocketTimeoutException {
368             MailjetRequest request = new MailjetRequest(Contact.resource).property(Contact
                .NAME, nomContact).property(Contact.EMAIL, mailContact);
369             MailjetResponse response = client.post(request);
370             if (response.getStatus() == 201)
371                 System.out.println("the contact is create");
372             else
373                 System.out.println("error");
374         }
375 
376         public static void addProperties(Integer ID, String Prenom, String type) throws
        MailjetException, MailjetSocketTimeoutException {
377             MailjetRequest request;
378             request = new MailjetRequest(Contactdata.resource, ID).property(Contactdata.
                DATA,
379                     new JSONArray().put(new JSONObject().put("Name", type).put("Value",
                        Prenom)));
380             client.put(request);
381         }
382 
383         // this function can serach a proprieties title in a json and catch the data
384         // in
385         public static String searchAContact(Object search) throws MailjetException,
        MailjetSocketTimeoutException {
386             String str = "";
387             MailjetRequest request = new MailjetRequest(Contact.resource).filter("limit",
                0);
388             MailjetResponse response = client.get(request);
389             if (isValidEmailAddress(search.toString()) == true) {
390                 for (int i = 0; i < response.getData().length(); i++) {
391                     if (search.equals(response.getData().getJSONObject(i).get("Email")))
                        str += response.getData().getJSONObject(i).get("ID");
392                 }
393             } else
394                 System.out.println("Error, retry");
395             return str;
396         }
397 
398         public static String searchAContactWithId(Integer ID) throws MailjetException,
        MailjetSocketTimeoutException {
399             MailjetRequest request = new MailjetRequest(Contact.resource, ID);
400             MailjetResponse response = client.get(request);
401             return response.getData().getJSONObject(0).get("Email").toString();
402         }
403 
404         private static List<?> getObjectListFromObject(final String json, final Class<?>
        className) throws JsonProcessingException, JsonMappingException {
405             try {
406                 final ObjectMapper objectMapper = new ObjectMapper().configure(
                    DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false);
407                 JavaType customClassCollection = objectMapper.getTypeFactory().
                    constructCollectionType(List.class, className);
408                 return objectMapper.readValue(json, customClassCollection);
409             } catch (Exception e) {
410                 e.printStackTrace();
411             }
412             return null;
413         }
414 
```

```
415     public static boolean isValidEmailAddress(String email) {
416         boolean result = true;
417         try {
418             InternetAddress emailAddr = new InternetAddress(email);
419             emailAddr.validate();
420         } catch (AddressException ex) {
421             result = false;
422         }
423         return result;
424     }
425 
426     public static String convertDate(Date uneDate) {
427         SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd", Locale.
        getDefault());
428         return (simpleDateFormat.format(uneDate).toString());
429     }
430 }
```