

Building a Mail Merge over a Hand-Written String Type

Final project — Programming I

Universitat de les Illes Balears

Authors:

Andreu Massanet Felix
Marc Morlà Maimó

2023 - 2024

Contents

1	Introduction	2
2	Design	2
2.1	The main program	2
2.1.1	Structure	2
2.1.2	Generating letters	2
2.2	Managing the address book	3
2.2.1	Adding a contact	3
2.2.2	Consulting contacts	3
2.2.3	Showing contacts	3
2.2.4	Deleting a contact	3
2.3	Managing distribution lists	3
2.3.1	administrarListas	4
2.3.2	creaLista	4
2.3.3	eliminaLista	4
2.3.4	seleccionaLista	4
2.4	Data structures	4
3	User manual	5
3.1	Generating letters	5
3.2	Managing the address book	5
3.3	Managing distribution lists	5
3.4	Notes on how it behaves	5
4	Conclusions	6

1 Introduction

This project is a program that prepares personalised letters from an address book and a document template written in HTML. It is driven from the command line through menus, so the user picks a task or leaves; once a task finishes the menu comes back, and the program does not end until the user says so.

Three features were implemented:

- **Generating letters:** the user gives the batch a title, chooses a template, chooses either a distribution list or the whole address book, and gets one personalised copy of the template per contact.
- **Managing the address book:** adding contacts, listing them all, searching by name or surname, searching by email address, and deleting a contact by email. Deletion asks for confirmation first, and removes the contact from every distribution list as well as from the book.
- **Managing distribution lists:** creating a list, deleting a list, and selecting one in order to add contacts to it or remove them from it.

The classes written for it are `Contacto`, `ContactoFicherosLectura` and `ContactoFicherosEscritura`, `Linea`, `LineaFicherosLectura` and `LineaFicherosEscritura`, and finally `main`, which holds most of the behaviour.

What makes the exercise more than a short script is the constraint it was set under: the string handling could not lean on Java's own `String` methods. Searching for a marker inside a line, cutting the line around it and splicing a value into the gap all had to be built from characters upwards, and that is what `Linea` is.

2 Design

A series of classes implement both the functions and the procedures the program needs.

2.1 The main program

2.1.1 Structure

There is a `main()` method, which is the entry point, and a `menu()` method built around a switch that shows the main menu and its options.

2.1.2 Generating letters

A do-while loop keeps this section running until the user chooses to leave.

Interaction with the user. The letter-generation menu is shown. The user is asked for a name for the batch, and a directory with that name is created under `fitxers/generats/`. The user is then asked for the name of the template.

Checking what exists. The template is looked for at `fitxers/plantilles/` followed by the name and `.html`. If it is not there, a message is printed and the directory that had just been created is removed again. The same happens if a batch of that name already exists.

Choosing the recipients. The user either names a distribution list or asks for the whole address book.

The whole address book. A loop walks every contact in `fitxers/agenda.txt` and writes a personalised HTML file for each one at `fitxers/generats/`, then the batch name, then the contact's email address and `.html`. The template is read and its markers are replaced with that contact's data.

A specific distribution list. The list is looked for at `fitxers/l1istes/` followed by the name and `.txt`. If it exists, letters are generated for each contact in it by the same logic as for the whole book. If it does not, a message is printed and the directory is removed.

2.2 Managing the address book

A do-while loop keeps this section running until the user chooses to leave.

2.2.1 Adding a contact

Uses the static method `Contacto.Crea()` to build a contact from what the user types, and `ContactoFicherosEscritura` to write it to `fitxers/agenda.txt`.

2.2.2 Consulting contacts

Presents a menu for querying the book: show every contact, show contacts by name or surname, show contacts by email address, and leave.

2.2.3 Showing contacts

`mostrarContactos`, `mostrarContactosNombreApellido` and `mostrarContactoEmail` all use `ContactoFicherosLectura` to read from `fitxers/agenda.txt`. Depending on the criterion given, each checks whether the lines obtained through `contactoToLinea` contain what was asked for. `mostrarContactos` simply loops and shows them all.

2.2.4 Deleting a contact

Asks the user to confirm before deleting. It finds the contact the same way `mostrarContactoEmail` does, and verifies the outcome with a boolean.

Because a line cannot be removed from a text file in place, deletion works by copying: a loop reads through the file with `ContactoFicherosLectura` and writes every line except the matching one to a temporary file with `ContactoFicherosEscritura`, and the copy then replaces the original. This is applied to the address book and to the distribution lists alike, so no list is left pointing at a contact that no longer exists.

2.3 Managing distribution lists

A do-while loop keeps this section running until the user chooses to leave.

2.3.1 administrarListas

Shows a menu for managing the lists, with options to create one, delete one, select one, and return to the main menu. Each option calls its own method: `creaLista`, `eliminaLista` or `seleccionaLista`.

2.3.2 creaLista

Asks the user for the name of the list and creates a file at `fitxers/llices/`, the name, and `.txt`, checking first that it does not already exist. It uses `LineaFicherosEscritura` to write to the file and then closes it.

2.3.3 eliminaLista

Asks for the name of the list and checks whether the file exists at `fitxers/llices/`. If it does, the file is deleted; if it does not, the program reports that the list does not exist.

2.3.4 seleccionaLista

Asks for the name of the list and checks the same path. If the list exists, a menu is presented for adding contacts to it or removing them. It uses `ContactoFicherosLectura` to read contacts from `fitxers/agenda.txt` and `LineaFicherosEscritura` to modify the list.

2.4 Data structures

The data lives in files: the address book, the HTML templates and the distribution lists.

`Contacto` and `Linea` represent and manipulate that data. `Linea` is the one the rest is built on. It holds a fixed array of characters and a length rather than a `String`, and implements every operation the program needs on top of it:

- `adicionCaracter` and `adicionString` to append.
- `detectarCaracter` to find a character from a given position.
- `extraccionSubLinea` to cut a range out as a new line.
- `contiene` and `esIgual` to compare — the second is what deleting a contact by email relies on.
- `sustituirSublinea` to find a sub-line and replace it with another, which need not be the same length. This is the operation the whole merge rests on.
- `hayLinea` to report whether there is anything left to read.

`Contacto` holds four of these — name, surname, email address and phone number — and converts between a contact and a single stored line through `contactoToLinea` and `LineaToContacto`, using `#` as the separator. The same character marks the placeholders in the templates, so one delimiter serves both the storage format and the template format.

3 User manual

The program is opened in NetBeans, which is the environment it was built in. Running it shows a main menu:

1. **Generate letters:** produces personalised letters from an HTML template and the data held in the address book.
2. **Manage the address book:** adding, consulting and deleting contacts.
3. **Manage distribution lists:** creating, deleting and selecting lists, and adding contacts to them or removing them.
4. **Exit.**

3.1 Generating letters

1. **Name the batch:** the name of a new folder where the generated letters will be kept.
2. **Choose the template:** an HTML file from `fitxers/plantilles/`.
3. **Choose a distribution list, or the whole address book.**

3.2 Managing the address book

1. **Add a contact:** name, surname, email address and phone number.
2. **Consult contacts:** show them all, search by name or surname, or search by email address.
3. **Delete a contact:** by email address. The contact is also removed from every distribution list it appears in.

3.3 Managing distribution lists

1. **Create a list,** by name.
2. **Delete a list,** by name.
3. **Select a list,** in order to add contacts to it or remove them.

3.4 Notes on how it behaves

Files used. Contacts are held in `fitxers/agenda.txt`, the HTML templates in `fitxers/plantilles/`, and the distribution lists in `fitxers/llices/`.

Feedback. The program reports what it has done — a contact added or removed, letters generated — and whether it succeeded.

Confirmation. Anything that modifies an important file, such as deleting a contact, asks the user to confirm first.

Errors. Messages are given for the situations that can be anticipated: a file that already exists where one is about to be created, a template that cannot be found, and options that are not valid.

4 Conclusions

The program does everything that was asked of it — generating letters, managing an address book, managing distribution lists — and building it made clear that programming at this level, taking a project of this size from nothing, is not a small task.

It was also a motivating one, precisely because parts of it did not come easily. Deleting a contact was the clearest example: it looks trivial until you find that a line cannot be taken out of a text file in place, and the answer turns out to be rewriting the file around it.

Above all, the exercise put the whole course into practice at once, and made the role programming plays a good deal more concrete than any single lecture could.