MEMORIA DE LA PRÁCTICA

PROGRAMACIÓN I

Nombres: **Felix Aguilar** y **Stefan Dimitrov**
Documentos de identificación: **43202115L** y **X4556925G**
Fecha de entrega: **miércoles 03/02/2016**
Nombres de los profesores: **Cristina Manresa** y **Miquel Mascaró Portells**
Grupo: **03**

Índice

Portada ……..... página 1

Índice ……... página 2

Introducción …….. página 3

Resumen ……... página 4

Explicación de las diferentes clases ……..…….. página 5

Manual de usuario ……... página 6

Imágenes de los problemas ……... páginas 7, 8 y 9

Conclusión ……...……. página 10

Introducción:

Nuestro proyecto está enfocado para la utilización en empresas, su función es escribir correos para los clientes, los cuales están previamente incorporados en el programa, siguiendo un mismo esquema pero puede usar diferentes cuerpos para los correos.
Tenemos 2 ficheros de entrada, uno con el listado de cartas (cada carta en una línea) que hay que imprimir. Cada línea contiene el tipo de carta que és (más adelante explicaremos esto), y el código del cliente al que hay que enviarla.
El segundo fichero contiene los clientes, cada uno con su código, el nombre y la dirección.
Después, hay 3 ficheros, uno para cada tipo de carta (en total 3 tipos, pues).

Resumen:

El trabajo está dividido en 5 clases cada una con sus subprogramas y variables.

Las clases son: la clase principal, la clase palabra, la clase fichero, la clase carta y la clase cliente. La clase palabra será usada para las operaciones de lectura de palabras y las operaciones con líneas (siempre se le entrará una array de chars y después la clase de eso hará palabras individuales, cada una con su longitud). La clase fichero abrirá los ficheros de lectura y escritura, y creará los ficheros indicados por el fichero de entrada (según és explicado en la introducción y más adelante). La clase carta és la clase que se encarga de crear las clases indicadas en el fichero de entrada, y la clase cliente será la que gestiona los datos del cliente al que se envía la carta. Allí también se leerá el código de cada cliente del fichero de clientes.

Primero de todo, se realiza la presentación y se da la bienvenida. Después, abrimos el fichero de entrada (leemos la primera línea) y obtenemos el código del cliente al que tiene que ser enviada la carta. Después, abrimos el fichero de clientes y buscamos el código del cliente en concreto. Si se encuentra, creamos la carta (membrete empresa, datos del cliente, fecha, y el tipo de carta que también está indicado en el fichero de entrada). Sinó, se explica por pantalla cuál cliente no ha sido encontrado. Y después todo de nuevo con la segunda línea del fichero de entrada, hasta que se acaba.
Todo esto va creando ficheros en la carpeta del proyecto, cada uno llamado con el nombre del cliente, y contiene el membrete empresa, datos del cliente, fecha, y el tipo de carta.

Si hay una excepción del tipo IO (lo más común es que no se encuentre alguno de los ficheros de entrada, o uno de los tipos de carta), explica por pantalla el error más probable.

Explicación de las diferentes clases:

El trabajo está dividido en 5 clases cada una con sus subprogramas y variables.

Las clases son:

Principal: En esta clase está el subprograma Inicio, en el qual esta las instrucciones que nos permiten usar los subprogramas que están en otras clases como aquellos usados para leer ficheros o la clase palabra, además en esta clase está el inicio del programa entero, en el cual está solamente la llamada del subprograma Inicio.

Cliente: Esta clase está especificada para obtener y gestionar a los clientes, en esta hay subprogramas que nos permiten lo dicho anteriormente mencionado, además, está la variable cliente, en la qual hay estos rasgos:

-Código: Es el codigo del cliente, cada cliente tiene un código distinto usado para diferenciarse del resto.

-Nombre: Es el nombre que pondremos como título al archivo generado, además se usa para escribirlo en la carta.

-Dirección: Es la dirección del cliente y solo se usa para escribirla en la carta juntamente con el nombre.

Carta: En esta clase, hay la variable carta, la qual es usada para saber que tipo de carta vamos a usar, ya que hay tres tipos:

-De bienvenida

-De pedido

-De catálogo

Además está el subprograma que nos permite saber que tipo de carta es la que necesitaremos con cada cliente.

Palabra: Una de las clases más importantes del programa, en esta esta la variable palabra, la cual se usa para poder editar y obtener lo que necesitamos en cada momento.

Además están los subprogramas necesarios para realizar estas tareas, tales como leer la palabra de la línea y saltar los espacios (estas dos trabajan conjuntamente para leer las palabras por separado) también está el subprograma “palabras iguales”, el cual está diseñado para saber si las palabras son iguales o no (en este caso se usa para saber si los códigos de la entrada y el cliente coinciden).

Fichero: Última clase de nuestro programa, la cual está diseñada para todo el trabajo de ficheros (sea escribir o leer de estos), En esta clase están los subprogramas que permiten lo dicho anteriormente, leer línea, crear fichero y escribir carta :

-leerLinea: utilizado para poder leer linea a linea un fichero

-crearFichero: utilizado para poder escribir la carta al cliente

-escribirCarta: utilizado para poder extraer el cuerpo de la carta.

Manual del usuario:

El programa Práctica está diseñado para que al iniciarse con los diferentes archivos proporcionados, genere diferentes archivos cada uno estará designado con el nombre del cliente.

Para empezar se han de proporcionar los diferentes archivos los cuales son el fichero de entrada:

El fichero Entrada.txt donde está el código del cliente y el tipo de carta que se quiere obtener para el.

Ej: 0000 Tipo de carta

El fichero Cliente.txt donde están los diferentes clientes con sus datos

Ej: 0000 #n nombre apellidos #d dirección del cliente

Con estos dos archivos, tiene todo lo necesario para poder ejecutar el programa, si falta alguno de los dos archivos el programa se parará con una excepción.

Con los dos archivos introducidos, el programa compara los códigos del cliente con los de la entrada para saber qué cliente va con esa instrucción de entrada. Si encuentra alguna compatibilidad

Ej: 0000 Tipo de carta = 0000 #n nombre apellidos #d dirección del cliente

entonces nombre apellidos necesita una carta del tipo Tipo de carta.

entonces se crea un fichero que se llama igual que el nombre del cliente en el cual se escribirá lo siguiente:

EMPRESA Zombotech
Av. Alexandre Rossello, 20
07002 PALMA DE MALLORCA
NIF: F12345678

Nombre del cliente
Dirección del cliente

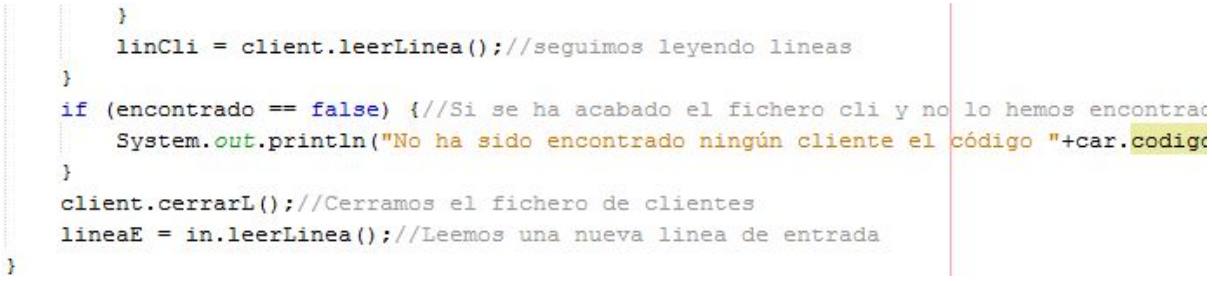
Fecha Actual del sistema

Cuerpo: hay diferentes cuerpos, cada uno es diferente, actualmente hay 3, tal y como se indicaba en el enunciado de la práctica.

Bienvenida
Catálogo
Pedido

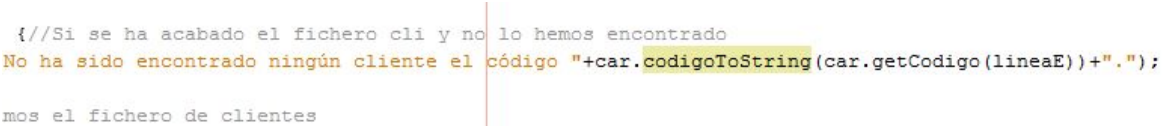
Problemas encontrados

Cada uno explicado con una imágen.



```
        }
        linCli = client.leerLinea();//seguimos leyendo lineas
    }
    if (encontrado == false) {//Si se ha acabado el fichero cli y no lo hemos encontrac
        System.out.println("No ha sido encontrado ningún cliente el código "+car.codigo
    }
    client.cerrarL();//Cerramos el fichero de clientes
    lineaE = in.leerLinea();//Leemos una nueva linea de entrada
}
```

En vez de encontrado == false, pusimos encontrado = false, lo que causaba que el if nunca era cierto, y por lo tanto nunca entraba en esa parte del programa. En consecuencia, ignoraría cuando el cliente del fichero de entrada no se encuentre en el fichero de clientes, lo que no es correcto, tiene que ser explicado.



```
 {//Si se ha acabado el fichero cli y no lo hemos encontrado
No ha sido encontrado ningún cliente el código "+car.codigoToString(car.getCodigo(lineaE))+".");

mos el fichero de clientes
```

En esas mismas líneas de código, explicando que el cliente no había sido encontrado, hemos decidido imprimir por pantalla el código de cliente que está en el fichero de entrada pero no en el de clientes, y lo hicimos de la forma car.getCodigo(con sus parámetros). Eso fue un error, ya que getCodigo de la clase Carta devuelve una objeto de clase palabra, y cuando intentamos imprimirlo, nos salía por pantalla palabra @ y después unos números aleatorios. Por lo tanto tuvimos que crear un nuevo método que transforma el código del cliente (de tipo Palabra) en un String, para poder imprimirlo.

```
public void leerPalabra(char[] lin, int ind) {//Leemos una palabra, nos pasan un array de chars y un índice desde el cual empeza
    idx = ind;
    saltarEspacios(lin);//Nos saltamos todos los posibles espacios
    for (longitud = 0; idx < lin.length && longitud < letras.length && lin[idx] != ' '; idx++, longitud++) {//Hasta que no se ac
        letras[longitud] = lin[idx];
    }
}

public void saltarEspacios(char[] linea) {
    while (linea[idx] == blanc && idx < linea.length) {//Mientras sea espacio, seguimos leyendo
        idx++;
    }
}
```

Aquí fue nuestro error más “grave” (en el sentido que no podíamos encontrarlo, hasta que nos ayudó una tutoría), y constaba que nos daba el error ArrayOutOfBounds, a causa de que leíamos la Palabra hasta que se acababa el array letras[], pero la línea podía ser más corta (el error mencionado arriba) o más larga (nos comíamos la parte de la línea después de la posición 20). Eso lo corregimos cambiando la condición del for a linea.length, en vez de letras.
Lo mismo pasaba con el método saltarEspacios, intentábamos saltarnos los espacios en el array letras[], pero como ese array es el que contiene la palabra, nunca tiene espacios, y en consecuencia no se saltaba en realidad los espacios de la línea. Después pasaba que se quedaba en un bucle infinito, porque siempre nos situamos en un espacio (por lo tanto no leíamos palabras), pero tampoco nos lo saltamos, porque el método que los saltaba era incorrecto. Pero por suerte, con ayuda de fuera, lo conseguimos arreglar.

```
public char[] leerLinea() throws IOException {//Leemos linea del fichero
    char[] lin = null;
    String linea = " ";
    linea = reader.readLine();
    if (linea != null) {//Si la linea no es vacia, hace una array de chars de la linea
        lin = linea.toCharArray();
    }
    return lin;
}
```

Aquí hubo otro error que nos costó averiguar. Nos daba continuamente una NullPointerException, lo que nos sorprendía y desquiciaba mucho, ya que en la clase principal teníamos que si la línea era null, salía y no hacía nada con ella. El error estaba en la clase Fichero, ya que al leer una línea, la convertimos de String a una array de chars, y cuando intentaba convertir un null en un array, daba error. Lo arreglamos con un if.

```
public void cerrarL() throws IOException {//Cerramos el fichero, si hemos leido
    reader.close();
    input.close();
}

public void cerrarE() throws IOException {//Cerramos el fichero, si hemos escrito
    writer.close();
    output.close();
}
```

Aquí nos daba también un NullPointerException, ya que teníamos un único método cerrar(), que cerraba tanto input y reader, como writer y output. Y al intentar cerrar algo que era null, daba un error. Lo arreglamos dividiéndolos en 2 métodos diferentes.

```
public String escribirCarta(Carta card) throws IOException {//Escribimos la carta
    String nom = "";
    for (int ind = 0; ind < card.getTipo().getLongitud(); ind++) {//Sacamos el tip
        nom = nom + card.getTipo().getLetras()[ind];
    }
    nom = nom + ".txt";//Lo ponemos como un array
    Fichero fcarta = new Fichero(nom, true);//Abrimos el fichero que tiene como no
    String carta = "";
    int valor = fcarta.reader.read();
    while (valor != -1) {
        char c = (char) valor;
        carta = carta + c;
        valor = fcarta.reader.read();//Leemos el membrete de la carta entero
    }
    fcarta.cerrarL();//Cerramos el fichero del membrete de la carta
    return carta;//Devolvemos el membrete de la carta como string, para poder dire
}
```

Aquí tuvimos que utilizar todos nuestros conocimientos sobre clases y POO para averiguar cómo había que hacerlo. Pensábamos que como estábamos en la clase fichero, se ponía directamente reader.read(). Pero no nos dimos cuenta (hasta más adelante) que estábamos en la clase fichero gracias al objeto de tipo fichero “salida”, que era el que escribía el fichero de salida. Por lo tanto, al intentar leer cuando el reader y el input no estaban inicializados, daba error. Averiguamos, después de un tiempo, que lo que tenemos que leer es otro objeto Fichero, en concreto fcarta, por lo tanto añadimos delante del reader.read() la llamada a ese objeto y ya todo funcionó como debía.

Conclusión:

Hacer el programa ha sido muy interesante, ya que hemos aprendido cómo son los sistemas de mensajería automatizada de empresas de internet, algo que siempre nos ha intrigado saber.

Por otra parte, hemos tenido bastantes problemas con el programa sobre todo con la lectura y escritura de ficheros, algunos de ellos los hemos sabido arreglar pero otros no, para poder resolver los últimos, hemos tenido que solicitar ayuda al profesor de prácticas la cual ha sido de mucha ayuda.

En conclusión el trabajo ha sido muy divertido de hacer a pesar de las complicaciones que hemos tenido y sin duda alguna nos ha ayudado a mejorar en esta asignatura.