Modificación Patrón Template

Objetivo

Usando el patrón Template, en el ejercicio de carga de pagos, vamos a agregar una nueva funcionabilidad.

Agreguemos un nuevo archivo que nos permita recibir los pagos realizados por medio de cargos automáticos, este archivo nos lo envía el banco y contiene los cargos realizados a los clientes directamente sobre su tarjeta de crédito/débito. El formato del archivo será en XML y tendrá la estructura que más te guste. El procesamiento y el log deberán tener las reglas ya conocidas.

Resolución

A continuación, se detallarán los pasos realizados para completar el objetivo de la tarea

1. Creación del archivo que enviara el banco en formato XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payments>
    <payment>
        <id>001</id>
        <customer>01</customer>
        <amount>100</amount>
        <date>10/07/2022</date>
    </payment>
    <payment>
        <id>002</id>
        <customer>02</customer>
        <amount>050</amount>
        <date>12/07/2022</date>
    </payment>
    <payment>
        <id>003</id>
        <customer>03</customer>
        <amount>090</amount>
        <date>13/07/2022</date>
    </payment>
    <payment>
        <id>004</id>
        <customer>10</customer>
        <amount>150</amount>
        <date>13/07/2022</date>
    </payment>
    <payment>
        <id>005</id>
        <customer>11</customer>
        <amount>200</amount>
        <date>14/07/2022</date>
    </payment>
</payments>
```

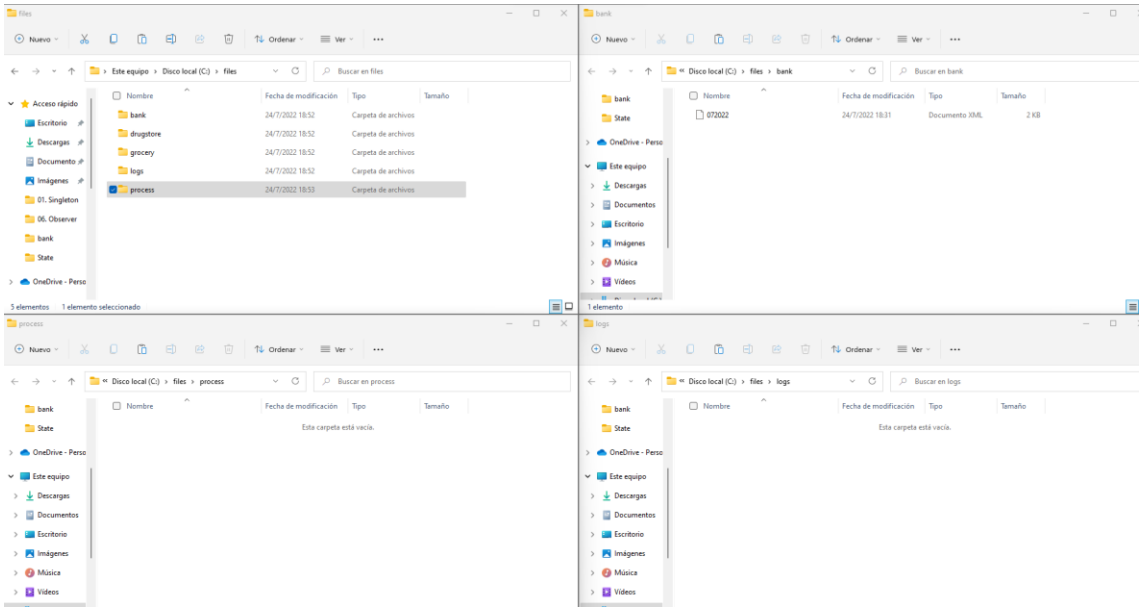
2. Creamos la clase BankFileProcess que es la clase que va a tener la validación del nombre de archivo, el método de procesar el archivo y el método de creación de log

```
public class BankFileProcess extends AbstractFileProcessTemplete {
    private String log = "";

    public BankFileProcess(File file, String logPath, String movePath){
        super(file, logPath, movePath);
    }

    @Override
    protected void validateName() throws Exception {
        String fileName = file.getName();
        if (!fileName.endsWith(".xml")) {
            throw new Exception("Invalid file name"
                    + ", must end with .xml");
        }

        if (fileName.length() != 10) {
            throw new Exception("Invalid document format");
        }
    }

    @Override
    protected void processFile() throws Exception {

        try {
            DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
            Document document = documentBuilder.parse(file);
            document.getDocumentElement().normalize();
            System.out.println("Root Element :" + document.getDocumentElement().getNodeName());
            NodeList nodeList = document.getElementsByTagName("payment");
            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);
                System.out.println("\nCurrent Element :" + node.getNodeName());
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element element = (Element) node;
                    String id = element.getAttribute("id");
                    String customer = element.getElementsByTagName("customer").item(0).getTextContent();
                    double amount = Double.parseDouble(element.getElementsByTagName("amount").item(0).getTextContent());
                    String date = element.getElementsByTagName("date").item(0).getTextContent();
                    boolean exist = OnMemoryDataBase.customerExist(
                            Integer.parseInt(customer));
                    if (!exist) {
                    log += id + " E" + customer + "\t\t" + date
                            + " Customer not exist\n";
                    } else if (amount > 200) {
                        log += id + " E" + customer + "\t\t" + date
                                + " The amount exceeds the maximum\n";
                    } else {

                        log += id + " E" + customer + "\t\t" + date
                                + " Successfully applied\n";
                    }
            System.out.println(e);
        }
    }

    @Override
    protected void createLog() throws Exception {
        FileOutputStream out = null;
        try {
            File outFile = new File(logPath + "/" + file.getName());
            if (!outFile.exists()) {
                outFile.createNewFile();
            }
            out = new FileOutputStream(outFile, false);
            out.write(log.getBytes());
            out.flush();
        } finally {
            out.close();
        }
    }
}
```

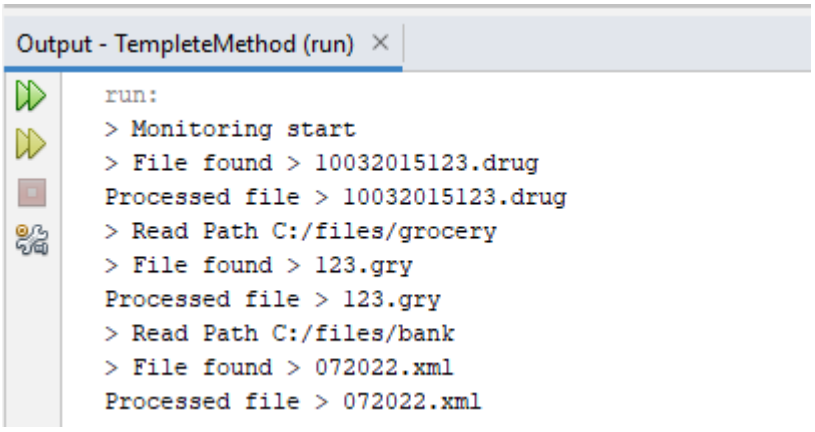
3. Modificamos el TempleteMethodMain donde agregamos la nueva ruta de los archivos bancarios y agregamos en el run las sentencias para que pueda leer la nueva ruta

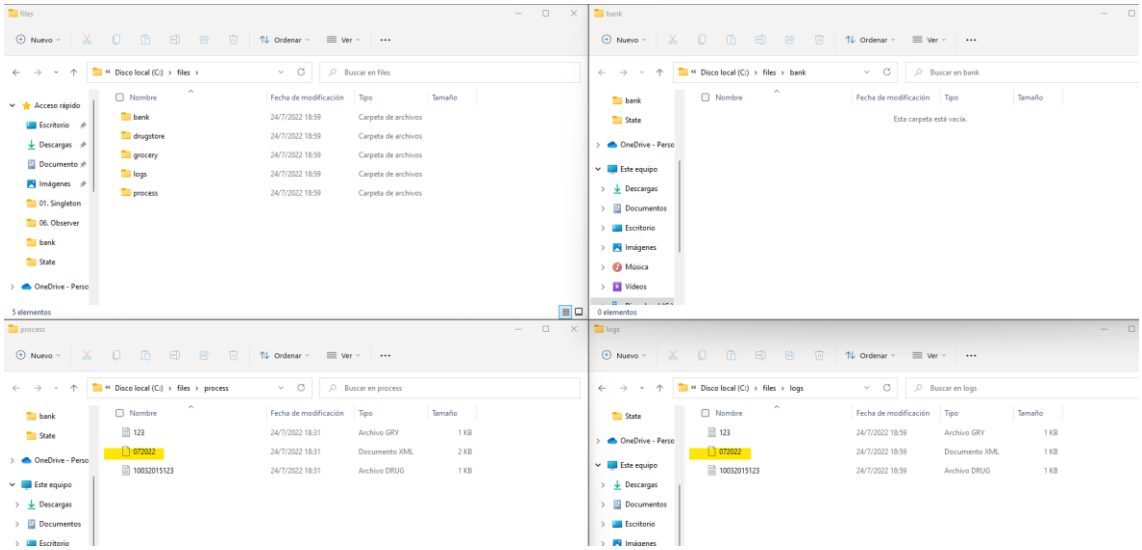
```java
public class TempleteMethodMain extends TimerTask {

    private static final String[] PATHS =
        new String[]{"C:/files/drugstore", "C:/files/grocery", "C:/files/bank"};
    private static final String LOG_DIR = "C:/files/logs";
    private static final String PROCESS_DIR = "C:/files/process";

    public static void main(String[] args) {
        new TempleteMethodMain().start();
    }

    public void start() {
        try {
            Timer timer = new Timer();
            timer.schedule(this, new Date(), (long) 1000 * 10);
            System.in.read();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public void run() {
        System.out.println("> Monitoring start");
        File f = new File(PATHS[0]);
        if(!f.exists()){
            throw new RuntimeException("El path '"+PATHS[0]+"' no existe");
        }
        File[] drugstoreFiles = f.listFiles();
        for (File file : drugstoreFiles) {
            try {
                System.out.println("> File found > " + file.getName());
                new DrugstoreFileProcess(file,LOG_DIR,PROCESS_DIR).execute();
                System.out.println("Processed file > " + file.getName());
            } catch (Exception e) {
                System.err.println(e.getMessage());
            }
        }

        f = new File(PATHS[1]);
        if(!f.exists()){
            throw new RuntimeException("El path '"+PATHS[1]+"' no existe");
        }
        System.out.println("> Read Path " + PATHS[1]);
        File[] groceryFiles = f.listFiles();
        for (File file : groceryFiles) {
            try {
                System.out.println("> File found > " + file.getName());
                new GroceryFileProcess(file,LOG_DIR,PROCESS_DIR).execute();
                System.out.println("Processed file > " + file.getName());
            } catch (Exception e) {
                System.err.println(e.getMessage());
            }
        }

        f = new File(PATHS[2]);
        if(!f.exists()){
            throw new RuntimeException("El path '"+PATHS[2]+"' no existe");
        }
        System.out.println("> Read Path " + PATHS[2]);
        File[] bankFiles = f.listFiles();
        for (File file : bankFiles) {
            try {
                System.out.println("> File found > " + file.getName());
                new BankFileProcess(file,LOG_DIR,PROCESS_DIR).execute();
                System.out.println("Processed file > " + file.getName());
            } catch (Exception e) {
                System.err.println(e.getMessage());
            }
        }
    }
}
```

4. Antes de correr el programa podemos observar que el archivo 072022 esta en la carpeta de bank y las carpetas de logs, process estan vacías



5. Una ves que corremos el programa se mueve el archivo de 072022 de bank a process y se crea un archivo de logs





6. El archivo logs del 072022 contiene lo siguiente después de haber pasado las validaciones

072022.xml 072022.xml

```
1   E01     10/07/2022 Successfully applied
2   E02     12/07/2022 Successfully applied
3   E03     13/07/2022 Successfully applied
4   E10     13/07/2022 Successfully applied
5   E11     14/07/2022 Customer not exist
6   E20     14/07/2022 Successfully applied
7   E25     14/07/2022 Customer not exist
8   E30     14/07/2022 Successfully applied
9   E40     15/07/2022 The amount exceeds the maximum
10  E50     16/07/2022 The amount exceeds the maximum
11
```

7. El código fuente se encuentra en la siguiente ruta https://github.com/cristianmoreno1986/patrones