

Modificación Patrón Template

Objetivo

Crear una nueva forma de pago, el cual se encarga de enviar el banco, sobre los cargos realizados a los clientes sobre su tarjeta de crédito/debito; dichos cargos deben ser leídos y procesados por el sistema; Estos archivos serán receptados en formato XML

Resolución

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

- i. Definición de la nueva estructura en XML;
Nombre del archivo: "[numero_cliente].xyz"

```
<?xml version="1.0"?>
<XyzData>
  <Documento>
    <Codigo>10024</Codigo>
    <Cliente>Cliente 10024</Cliente>
    <Fecha>2022-07-23</Fecha>
    <Producto>ITM_1</Producto>
    <Cantidad>5</Cantidad>
    <Precio>50</Precio>
    <Total>250</Total>
    <Observacion>Observacion del documento 1</Observacion>
  </Documento>
  <Documento>
    <Codigo>10024</Codigo>
    <Cliente>Cliente 10024</Cliente>
    <Fecha>2022-07-23</Fecha>
    <Producto>ITM_1</Producto>
    <Cantidad>1</Cantidad>
    <Precio>50</Precio>
    <Total>50</Total>
    <Observacion>Observacion del documento 1</Observacion>
  </Documento>
</XyzData>
```

- ii. Creacion del nuevo cliente sobre el archivo "OnMemoryDataBase"

```
public class OnMemoryDataBase {

    public static final Map<String, String> PROCESS_DOCUMENTS = new HashMap<>();
    public static final int[] CUSTOMERS = new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9,
        10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 10024};
    [ ... More code ... ]
}
```

- iii. Creación de la implementación del nuevo método de pago

```
public class EmpresaXYZFileProcess extends AbstractFileProcessTemplate {
    private String log = "";

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

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

```

    }

    @Override
    protected void processFile() throws Exception {
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        try {
            String fileName = this.file.getName();
            dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, true);

            // parse XML file
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document doc = db.parse(file);

            doc.getDocumentElement().normalize();
            System.out.println("Root Element :" +
            doc.getDocumentElement().getNodeName());
            System.out.println("-----");

            NodeList list = doc.getElementsByTagName("Documento");
            for (int temp = 0; temp < list.getLength(); temp++) {
                Node node = list.item(temp);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element element = (Element) node;
                    String codigo =
                    element.getElementsByTagName("Codigo").item(0).getTextContent();
                    String cliente =
                    element.getElementsByTagName("Cliente").item(0).getTextContent();
                    String fecha =
                    element.getElementsByTagName("Fecha").item(0).getTextContent();
                    String producto =
                    element.getElementsByTagName("Producto").item(0).getTextContent();
                    String cantidad =
                    element.getElementsByTagName("Cantidad").item(0).getTextContent();
                    String precio =
                    element.getElementsByTagName("Precio").item(0).getTextContent();
                    String total =
                    element.getElementsByTagName("Total").item(0).getTextContent();
                    String observacion =
                    element.getElementsByTagName("Observacion").item(0).getTextContent();

                    double totalAmount = Double.parseDouble(total);

                    boolean exist =
                    OnMemoryDataBase.customerExist(Integer.parseInt(codigo));
                    if (!exist) {
                        log += codigo + " E" + cliente + "\t\t" + fecha + " Customer
                        not exist\n";
                    } else if (totalAmount > 200) {
                        log += codigo + " E" + cliente + "\t\t" + fecha + " The amount
                        exceeds the maximum\n";
                    } else {
                        //TODO Aplicar el pago en algún lugar.
                        log += codigo + " E" + cliente + "\t\t" + fecha + "
                        Successfully applied\n";
                    }
                }
            }
        } finally { }
    }

    @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(); }
    }
}

```

- iv. Implementación del nuevo procesamiento para los archivos "*.xyz" ubicado en el archivo "TemplateMethodMain"

```
public class TemplateMethodMain extends TimerTask {

    private static final String[] PATHS =
        new String[]{"C:/files/drugstore", "C:/files/grocery",
"C:/files/empresaXYZ"};
    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 TemplateMethodMain().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");

        // Drugstore Files
        File f = new File(PATHS[0]);
        if(!f.exists()){ throw new RuntimeException("El path '"+PATHS[0]+"' no
existe"); }
        System.out.println("> Read Path " + PATHS[0]);
        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());
            }
        }

        // Grocery Files
        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());
            }
        }

        // Empresa XYZ Files
        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[] xyzFiles = f.listFiles();
        for (File file : xyzFiles) {
            try {
                System.out.println("> File found > " + file.getName());
                new EmpresaXYZFileProcess(file, LOG_DIR, PROCESS_DIR).execute();
                System.out.println("Processed file > " + file.getName());
            } catch (Exception e) { System.err.println(e.getMessage()); }
        }
    }
}
```

Patrón Template

Por medio de la clase `EmpresaXYZFileProcess` que esta implementa la clase abstracta `AbstractFileProcessTemplate` vamos a procesar los archivos con formatos XML; estos archivos deberán tener la extensión “xyz”

Entonces dentro de la clase principal `TemplateMethodMain`, agregamos la nueva ruta de trabajo y el respectivo bloque de código, para la lectura de los archivos que serán procesados, tal como se muestra en la sección de logs

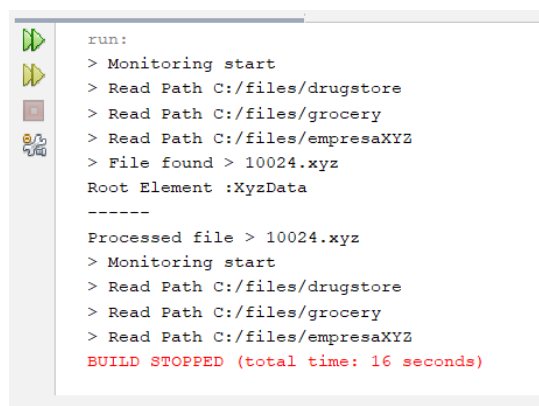
Gracias a la ayuda de este patrón, tal como en los ejemplos anteriores, solo se define los métodos relacionados a la validación del nombre del archivo, procesamiento del archivo (formato XML) y el guardado del log

Users

Resultado

Finalmente se tienen el siguiente resultado, con el nuevo Template de pago implementado

```
run:
> Monitoring start
> Read Path C:/files/drugstore
> Read Path C:/files/grocery
> Read Path C:/files/empresaXYZ
> File found > 10024.xyz
Root Element :XyzData
-----
Processed file > 10024.xyz
> Monitoring start
> Read Path C:/files/drugstore
> Read Path C:/files/grocery
> Read Path C:/files/empresaXYZ
BUILD STOPPED (total time: 16 seconds)
```



Resultado del archivo de log

10024	ECliente	10024	2022-07-23	The amount exceeds the maximum
10024	ECliente	10024	2022-07-23	Successfully applied
10024	ECliente	10024	2022-07-23	Successfully applied
10024	ECliente	10024	2022-07-23	The amount exceeds the maximum