Mr. Kundjanasith Thonglek
Student ID : 5710545023

Design Patterns with Java Spring

01219448 Software Patterns and Architecture

Instruction :

Calling using Spring Integration Enterprise Design Pattern using the library of Spring Integration

Required Installation :

1. Eclipse any version or Spring Tool Suite
2. Java Development Kit Version 7+
3. Spring Library
4. Given Files

Design for processing of bank invoices :

1. **Invoices Gateway** :
   Having a Gateway implemented as a component for sending invoice files into the system.

2. **Splitter** :
   Receiving the input invoice file, processing for each invoice - the messages from the invoice file to be split into for processing each message (invoice as the payload).

3. **Filter** :
   Processing invoices with the amount less than $10,000

4. **Router** :
   Sending messages with the invoices as the payload to the right channel. Some invoices will have the IBAN account ID with 2 bank types, I.e. domestic and cross-border transaction. The job on the Router component will send the message to the right channel.

5. **Transformers** :
   For transforming the data - firstly, receiving the invoice data, then changing the invoice object into the message object. Then sending to another Job module. Job in the transformer component will transform the message with the Invoice type to the payment payload.

6. **Banking Service Activator** :
   At the last step, the external module the Service Activator will call to by entering into the right channel.

Summary of components for Spring Integration :

Step : Gateway
File class component : InvoiceCollectorGateway
Class component as tag in XML file :

```
<int:gateway id="invoicesGateway"
                    service-interface="org.ku.invoices.InvoiceCollectorGateway">
        <int:method name="collectorInvoices" request-channel="newInvoicesChannel"/>
<int:gateway>
```

Step : Splitter
File class component : Channel
Class component as tag in XML file :

```
<int:splitter input-channel="newInvoicesChannel"
                                        output-channel="singleInvoicesChannel"/>
```

Step : Filter
File class component : InvoiceFilter
Class component as tag in XML file :

```
<int:filter input-channel="singleInvoicesChannel"
                    output- channel="filteredInvoicesChannel" ref="invoicesFilter"/>
```

Step : Router
File class component : Channel
Class component as tag in XML file :

```
<int:recipient-list-router input-channel="filteredInvoicesChannel"
        <int:recipient channel="foreignTransactions"
                selector-expression="payload.foreign"/>
        <int:recipient channel="localTransactions"
                selector-expression="payload.foreign"/>
</int:recipient-list-router>
```

Step : Transformer
File class component : ForeignPaymentCreator
Class component as tag in XML file :

```
<int:transformer input-channel="foreignTransactions"
                    output-channel="bankingChannel" ref="foreignPaymentCreator"/>
```

Step : Activator
File class component : PaymentProcessor
Class component as tag in XML file :

```
<int:service-activator input-channel="bankingChannel" ref="paymentProcessor">
        <int:poller fixed-rate="500" error-channel="failedPaymentsChannel"/>
</int:service-activator>
```

Application

```java
@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        ApplicationContext ctx = new AnnotationConfigApplicationContext(Config.class);
        InvoiceCollectorGateway gateway = ctx.getBean(InvoiceCollectorGateway.class);
        gateway.collectInvoices(MessageBuilder.withPayload(createInvoices()).build());

        ((AnnotationConfigApplicationContext) ctx).close();
    }

    private static List<Invoice> createInvoices() {
        List<Invoice> invoices = new ArrayList<Invoice>();

        invoices.add(new Invoice(InvoiceType.DOMESTIC, "current-local-acc", "test-account 9681000", 4706));
        invoices.add(new Invoice(InvoiceType.DOMESTIC, "current-local-acc", "test-account 5551000", 6531));
        invoices.add(new Invoice(InvoiceType.DOMESTIC, "current-local-acc", "test-account 9379135", 17083));
        invoices.add(new Invoice(InvoiceType.CROSS_BORDER, "current-iban-acc", "test-iban-2421000", 3601));
        invoices.add(new Invoice(InvoiceType.CROSS_BORDER, "current-iban-acc", "test-iban-2151059", 18761));
        invoices.add(new Invoice(InvoiceType.DOMESTIC, "current-local-acc", "test-account 2814014", 13286));
        invoices.add(new Invoice(InvoiceType.CROSS_BORDER, "current-iban-acc", "test-iban-9291000", 5765));
        invoices.add(new Invoice(InvoiceType.CROSS_BORDER, "current-iban-acc", "test-iban-4124989", 17192));
        invoices.add(new Invoice(InvoiceType.DOMESTIC, "current-local-acc", "test-account-2481040", 19139));
        invoices.add(new Invoice(InvoiceType.CROSS_BORDER, "current-iban-acc", "test-account-iban-7961000", 1699));

        return Collections.unmodifiableList(invoices);
    }


}
```

Configuration

```java
@SpringBootApplication
@ComponentScan(basePackages="com.spa.assignment4")
@IntegrationComponentScan(basePackages="com.spa.assignment4")
public class Config {
    @Bean
    public MessageChannel newInvoicesChannel() { return new DirectChannel(); }
    @Bean
    public MessageChannel singleInvoicesChannel() { return new DirectChannel(); }
    @Bean
    public MessageChannel filteredInvoicesChannel() { return new DirectChannel(); }
    @Bean
    public MessageChannel foreignTransactions() { return new DirectChannel(); }
    @Bean
    public MessageChannel localTransactions() { return new DirectChannel(); }
    @Bean
    public MessageChannel bankingChannel() { return new DirectChannel(); }
}
```

ForeignInvoiceMessage

```java
public class ForeignInvoiceMessage extends InvoiceMessage {
    public ForeignInvoiceMessage(Invoice invoice) { super(invoice); }
    @Override
    public String toString() { return "Foreign"+super.toString(); }
}
```

Invoice

```java
public class Invoice {
    public enum InvoiceType { DOMESTIC, CROSS_BORDER }
    private InvoiceType invoiceType;
    private String senderAccount;
    private String receiverAccount;
    private double amount;
    public Invoice(InvoiceType invoiceType, String senderAccount, String receiverAccount, double amount) {
        this.invoiceType = invoiceType; this.senderAccount = senderAccount;
        this.receiverAccount = receiverAccount; this.amount = amount;
    }
    public InvoiceType getInvoiceType() { return invoiceType; }
    public void setInvoiceType(InvoiceType invoiceType) { this.invoiceType = invoiceType; }
    public String getSenderAccount() { return senderAccount; }
    public void setSenderAccount(String senderAccount) { this.senderAccount = senderAccount; }
    public String getReceiverAccount() { return receiverAccount; }
    public void setReceiverAccount(String receiverAccount) { this.receiverAccount = receiverAccount; }
    public double getAmount() { return amount; }
    public void setAmount(double amount) { this.amount = amount; }
    @Override
    public String toString() {
        return "Invoice [invoiceType=" + invoiceType + ", senderAccount=" + senderAccount + ", receiverAccount="
                + receiverAccount + ", amount=" + amount + "]";
    }
    @Override
    public int hashCode() {
        final int prime = 31; int result = 1; long temp; temp = Double.doubleToLongBits(amount);
        result = prime * result + (int) (temp ^ (temp >>> 32));
        result = prime * result + ((invoiceType == null) ? 0 : invoiceType.hashCode());
        result = prime * result + ((receiverAccount == null) ? 0 : receiverAccount.hashCode());
        result = prime * result + ((senderAccount == null) ? 0 : senderAccount.hashCode());
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true; if (obj == null) return false;
        if (getClass() != obj.getClass()) return false; Invoice other = (Invoice) obj;
        if (Double.doubleToLongBits(amount) != Double.doubleToLongBits(other.amount)) return false;
        if (invoiceType != other.invoiceType) return false;
        if (receiverAccount == null) { if (other.receiverAccount != null) return false;
        } else if (!receiverAccount.equals(other.receiverAccount)) return false;
        if (senderAccount == null) { if (other.senderAccount != null) return false;
        } else if (!senderAccount.equals(other.senderAccount)) return false; return true;
    }
}
```

InvoiceMessage

```java
public class InvoiceMessage {
    private Invoice invoice;
    public InvoiceMessage(Invoice invoice) { this.invoice = invoice; }
    public Invoice getInvoice() { return invoice; }
    @Override
    public String toString() { return invoice.toString(); }
}
```

LocalInvoiceMessage

```java
public class LocalInvoiceMessage extends InvoiceMessage {
    public LocalInvoiceMessage(Invoice invoice) { super(invoice); }
    @Override
    public String toString() { return "Local"+super.toString(); }
}
```

Screen Capturing of the result

```
:35:26.633 [main] INFO org.springframework.integration.endpoint.EventDrivenConsumer - Removing {filter:invoiceFilter.filter
:35:26.633 [main] INFO org.springframework.integration.channel.DirectChannel - Channel 'org.springframework.context.annotat
:35:26.633 [main] DEBUG org.springframework.context.support.DefaultLifecycleProcessor - Bean 'invoiceFilter.filterInvoiceAm
:35:26.633 [main] INFO org.springframework.integration.endpoint.EventDrivenConsumer - stopped invoiceFilter.filterInvoiceAm
:35:26.633 [main] DEBUG org.springframework.context.support.DefaultLifecycleProcessor - Asking bean 'invoiceRouter.route.ro
:35:26.633 [main] INFO org.springframework.integration.endpoint.EventDrivenConsumer - Removing {router:invoiceRouter.route.
:35:26.633 [main] INFO org.springframework.integration.channel.DirectChannel - Channel 'org.springframework.context.annotat
:35:26.633 [main] DEBUG org.springframework.context.support.DefaultLifecycleProcessor - Bean 'invoiceRouter.route.router' c
:35:26.633 [main] INFO org.springframework.integration.endpoint.EventDrivenConsumer - stopped invoiceRouter.route.router
:35:26.633 [main] DEBUG org.springframework.context.support.DefaultLifecycleProcessor - Asking bean 'invoiceSplitter.splitM
:35:26.633 [main] INFO org.springframework.integration.endpoint.EventDrivenConsumer - Removing {splitter:invoiceSplitter.sp
:35:26.633 [main] INFO org.springframework.integration.channel.DirectChannel - Channel 'org.springframework.context.annotat
:35:26.633 [main] DEBUG org.springframework.context.support.DefaultLifecycleProcessor - Bean 'invoiceSplitter.splitMessage.
:35:26.633 [main] INFO org.springframework.integration.endpoint.EventDrivenConsumer - stopped invoiceSplitter.splitMessage.
:35:26.633 [main] DEBUG org.springframework.context.support.DefaultLifecycleProcessor - Asking bean 'paymentProcessor.proce
:35:26.633 [main] INFO org.springframework.integration.endpoint.EventDrivenConsumer - Removing {service-activator:paymentPr
:35:26.633 [main] INFO org.springframework.integration.channel.DirectChannel - Channel 'org.springframework.context.annotat
:35:26.633 [main] DEBUG org.springframework.context.support.DefaultLifecycleProcessor - Bean 'paymentProcessor.processInvoi
:35:26.633 [main] INFO org.springframework.integration.endpoint.EventDrivenConsumer - stopped paymentProcessor.processInvoi
:35:26.634 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Destroying singletons in org.
:35:26.634 [main] DEBUG org.springframework.beans.factory.support.DisposableBeanAdapter - Invoking destroy() on bean with n
:35:26.634 [main] INFO org.springframework.jmx.export.annotation.AnnotationMBeanExporter - Unregistering JMX-exposed beans
:35:26.634 [main] INFO org.springframework.jmx.export.annotation.AnnotationMBeanExporter - Unregistering JMX-exposed beans
:35:26.634 [main] DEBUG org.springframework.beans.factory.support.DisposableBeanAdapter - Invoking destroy() on bean with n
:35:26.634 [main] INFO org.springframework.scheduling.concurrent.ThreadPoolTaskScheduler - Shutting down ExecutorService 't
:35:26.635 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Retrieved dependent beans fo
:35:26.635 [main] DEBUG org.springframework.beans.factory.support.DisposableBeanAdapter - Invoking destroy() on bean with n
```