Shehab

DESIGN PATTERNS - CPIT252

LAB\_9

Contents

[Classes 2](#_Toc102438746)

[Payment 2](#_Toc102438747)

[CreditCardPayment 2](#_Toc102438748)

[PayPalPayment 3](#_Toc102438749)

[Receipt 3](#_Toc102438750)

[EmailReceipt 4](#_Toc102438751)

[PdfReceipt 4](#_Toc102438752)

[PrintReceipt 5](#_Toc102438753)

[Explain how the strategy design pattern simlified the process of adding a new payment and receipt delivery options? 6](#_Toc102438754)

[Output 6](#_Toc102438755)

# Classes

Text

Description automatically generated

I’ll only show the classes that has been modified.

## Payment

public interface Payment {  
 void pay(double amount);  
}

## CreditCardPayment

package edu.kau.fcit.cpit252.paymentsStrategy;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.UUID;  
  
public class CreditCardPayment implements Payment{  
 private String transactionId;  
 private String name;  
 private String cardNumber;  
 private String cvv;  
 private String monthYearExpiration;  
 private Date date;  
  
  
 public CreditCardPayment(String name, String cardNumber, String cvv,  
 String monthYearExpiration) {  
 this.transactionId = UUID.*randomUUID*().toString();  
 this.name = name;  
 this.cardNumber = cardNumber;  
 this.cvv = cvv;  
 this.monthYearExpiration = monthYearExpiration;  
 this.date = new Date();  
 }  
  
  
 @Override  
 public String toString() {  
 String datePattern = "dd-MM-yyyy";  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat(datePattern);  
 return "Credit Card Payment " +  
 "\n\tTransaction Id: " + this.transactionId +  
 "\n\tDate: " + simpleDateFormat.format(this.date) +  
 "\n\tCard Number: \*\*\*\*" +  
 this.cardNumber.substring(this.cardNumber.length() - 5, this.cardNumber.length() - 1);  
 }  
  
 @Override  
 public void pay(double amount) {  
 System.*out*.println("Payment has completed by Credit Card!");  
 System.*out*.println(this.toString());  
 }  
}

I made this class implement “**Payment**” interface. And override the method containing feed back for the user.

## PayPalPayment

package edu.kau.fcit.cpit252.paymentsStrategy;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.UUID;  
  
public class PayPalPayment implements Payment {  
 private String transactionId;  
 private String email;  
 public String merchant;  
 public Date date;  
  
 public PayPalPayment(String email, String merchant) {  
 this.email = email;  
 this.merchant = merchant;  
 this.date = new Date();  
 this.transactionId = UUID.*randomUUID*().toString();  
 }  
 @Override  
 public String toString() {  
 String datePattern = "dd-MM-yyyy";  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat(datePattern);  
 return String.*format*("PayPal Payment:" +  
 "\n\tTransaction Id " + this.transactionId +  
 "\n\tDate: " + simpleDateFormat.format(this.date) +  
 "\n\tMerchant: " + this.merchant +  
 "\n\t" + "Email: " + this.email + "\n");  
 }  
  
 @Override  
 public void pay(double amount) {  
 System.*out*.println("Payment has completed by PayPal!");  
 System.*out*.println(this.toString());  
 }  
}

Same as the “**CreditCardPayment**” class.

## Receipt

package edu.kau.fcit.cpit252.receiptStrategy;  
  
import edu.kau.fcit.cpit252.shopping.Product;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.List;  
import java.util.UUID;  
  
public abstract class Receipt {  
 private String id;  
 private List<Product> products;  
 private String issueDate;  
  
 public Receipt(List<Product> products){  
 this.id = UUID.*randomUUID*().toString();  
 this.products = products;  
 String datePattern = "dd-MM-yyyy";  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat(datePattern);  
 this.issueDate = simpleDateFormat.format(new Date());  
 }  
  
  
 public String getId() {  
 return id;  
 }  
  
 public List<Product> getProducts() {  
 return products;  
 }  
  
 public String getIssueDate() {  
 return issueDate;  
 }  
  
 public abstract void generate();  
  
}

I’ve made this class abstract and the method generate too, since it has no implementation and, the other classes in the same package do override it.

## EmailReceipt

package edu.kau.fcit.cpit252.receiptStrategy;  
  
import edu.kau.fcit.cpit252.shopping.Product;  
import edu.kau.fcit.cpit252.utils.EmailTemplate;  
import edu.kau.fcit.cpit252.utils.SendEmail;  
  
import java.util.List;  
  
public class EmailReceipt extends Receipt {  
 private String subject;  
 private String recipient;  
  
 public EmailReceipt(String subject, String recipient, List<Product> list) {  
 super(list);  
 this.subject = subject;  
 this.recipient = recipient;  
 }  
  
 @Override  
 public void generate() {  
 EmailTemplate template = new EmailTemplate();  
 String message = template.getEmailTemplate("Our Neighborhood Bakery", this.getId(), this.getIssueDate(),  
 this.getProducts());  
 try {  
 SendEmail.*send*(this.subject, message, this.recipient);  
 }  
 catch(Exception e){  
 System.*err*.println("Failed to email the receipt: Reason: " + e.getMessage());  
 return;  
 }  
  
 }  
}

I’ve made this class inherit “**Receipt**” class + generated a constructor passed on the constructor that was provided in the main method, and passed the list that is a parameter in the constructor to the super class “**Receipt**”.

## PdfReceipt

package edu.kau.fcit.cpit252.receiptStrategy;  
  
import edu.kau.fcit.cpit252.shopping.Product;  
import org.apache.pdfbox.pdmodel.PDDocument;  
import org.apache.pdfbox.pdmodel.PDDocumentInformation;  
import org.apache.pdfbox.pdmodel.PDPage;  
import org.apache.pdfbox.pdmodel.PDPageContentStream;  
import org.apache.pdfbox.pdmodel.font.PDType1Font;  
import java.io.IOException;  
import java.util.GregorianCalendar;  
import java.util.List;  
  
public class PdfReceipt extends Receipt{  
  
 private String subject;  
 private String path;  
  
 public PdfReceipt(String subject, String path, List<Product> list) {  
 super(list);  
 this.subject = subject;  
 this.path = path;  
 }  
  
 @Override  
 public void generate() {  
 StringBuilder sb = new StringBuilder();  
 sb.append("\t\t\t" + this.subject + "\n");  
 sb.append("-----------\t\t\t --------------\n");  
 sb.append("Item\t\t\t\t Price\n");  
 sb.append("-----------\t\t\t --------------\n");  
 double total = 0;  
 for (Product p : this.getProducts()) {  
  
 sb.append("\t" + p.getName() + "\t\t\t" + p.getPrice() + "\n");  
 total += p.getPrice();  
 }  
 sb.append("\t\t\t\t\t Total:\t" + total);  
  
 PDDocument document = new PDDocument();  
 PDDocumentInformation pdfDocInfo = document.getDocumentInformation();  
 PDPage pdfPage = new PDPage();  
 pdfDocInfo.setTitle(this.subject);  
 pdfDocInfo.setCreationDate(new GregorianCalendar());  
 try {  
 document.addPage(pdfPage);  
 PDPageContentStream contentStream = new PDPageContentStream(document, pdfPage);  
 contentStream.beginText();  
 contentStream.setFont(PDType1Font.*TIMES\_ROMAN*, 12);  
 contentStream.setLeading(14.5f);  
 contentStream.newLineAtOffset(25, 725);  
  
 String content = sb.toString().replaceAll("\t", " ");  
 String[]lines = content.split("\n");  
  
 for(String line : lines){  
 contentStream.showText(line);  
 contentStream.newLine();  
 contentStream.newLineAtOffset(0,-10);  
 }  
 contentStream.endText();  
 contentStream.close();  
 document.save(this.path);  
 document.close();  
 }  
 catch(IOException e){  
 System.*err*.println("Error: Failed to print a PDF receipt. Reason: " + e.getMessage());  
 }  
 }  
}

Same as “**EmailReceipt**” class.

## PrintReceipt

package edu.kau.fcit.cpit252.receiptStrategy;  
  
import edu.kau.fcit.cpit252.shopping.Product;  
  
import java.util.List;  
  
public class PrintReceipt extends Receipt {  
  
 private String subject;  
  
 public PrintReceipt(String subject, List<Product> products) {  
 super(products);  
 this.subject = subject;  
 }  
  
 @Override  
 public void generate() {  
 StringBuilder sb = new StringBuilder();  
 sb.append("\t\t\t" + this.subject + "\n");  
 sb.append("-----------\t\t\t --------------\n");  
 sb.append("Item\t\t\t\t Price\n");  
 sb.append("-----------\t\t\t --------------\n");  
 double total = 0;  
 for (Product p : this.getProducts()) {  
  
 sb.append("\t" + p.getName() + "\t\t\t" + p.getPrice() + "\n");  
 total += p.getPrice();  
 }  
 sb.append("\t\t\t\t\t Total:\t" + total);  
 System.*out*.println(sb.toString());  
 }  
}

Same as “**EmailReceipt**” class.

# Explain how the strategy design pattern simlified the process of adding a new payment and receipt delivery options?

Since it gave each payment method it’s own algorithm so as printing the receipt.

# Output

Text

Description automatically generated

Text

Description automatically generated