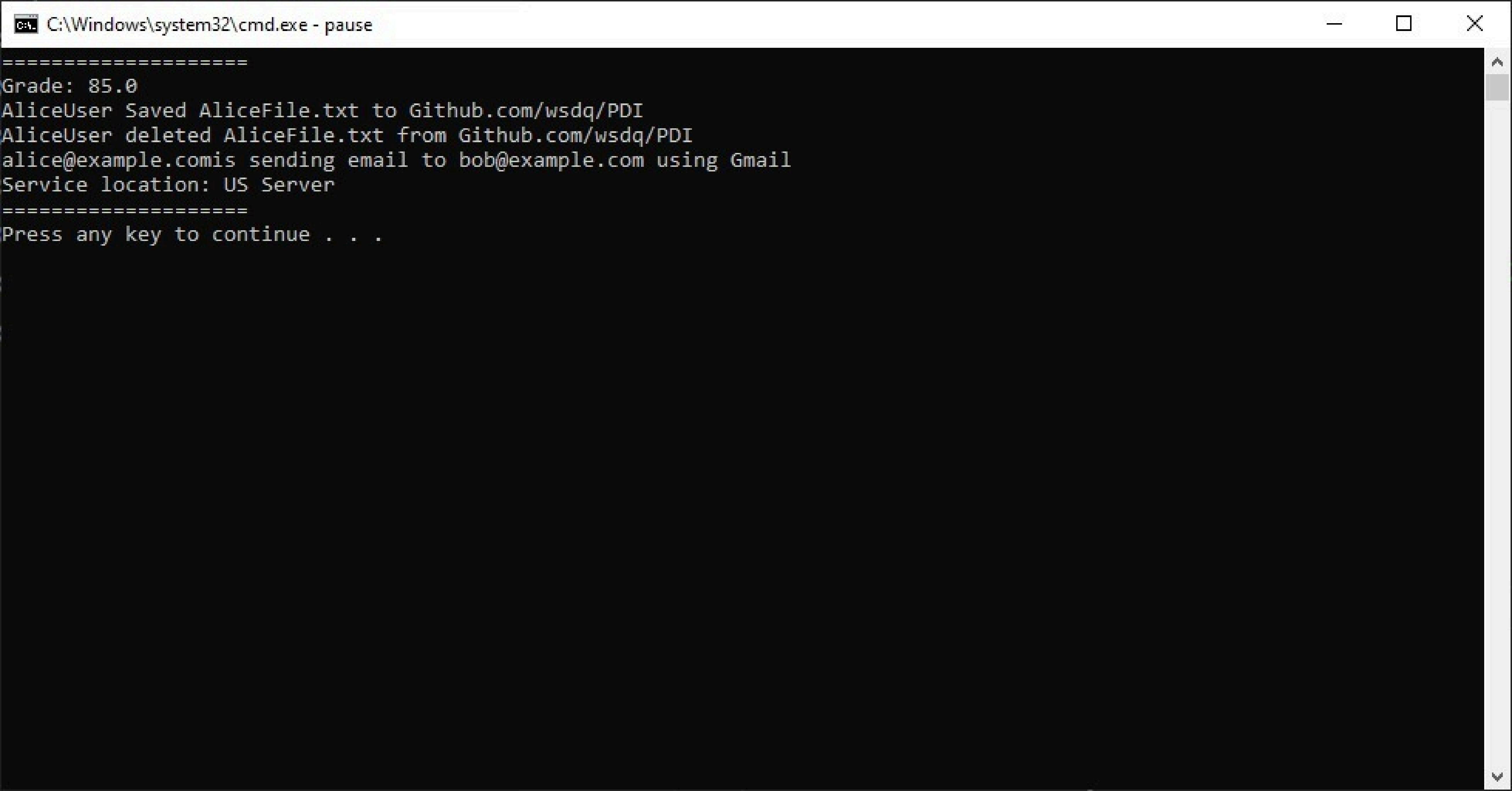
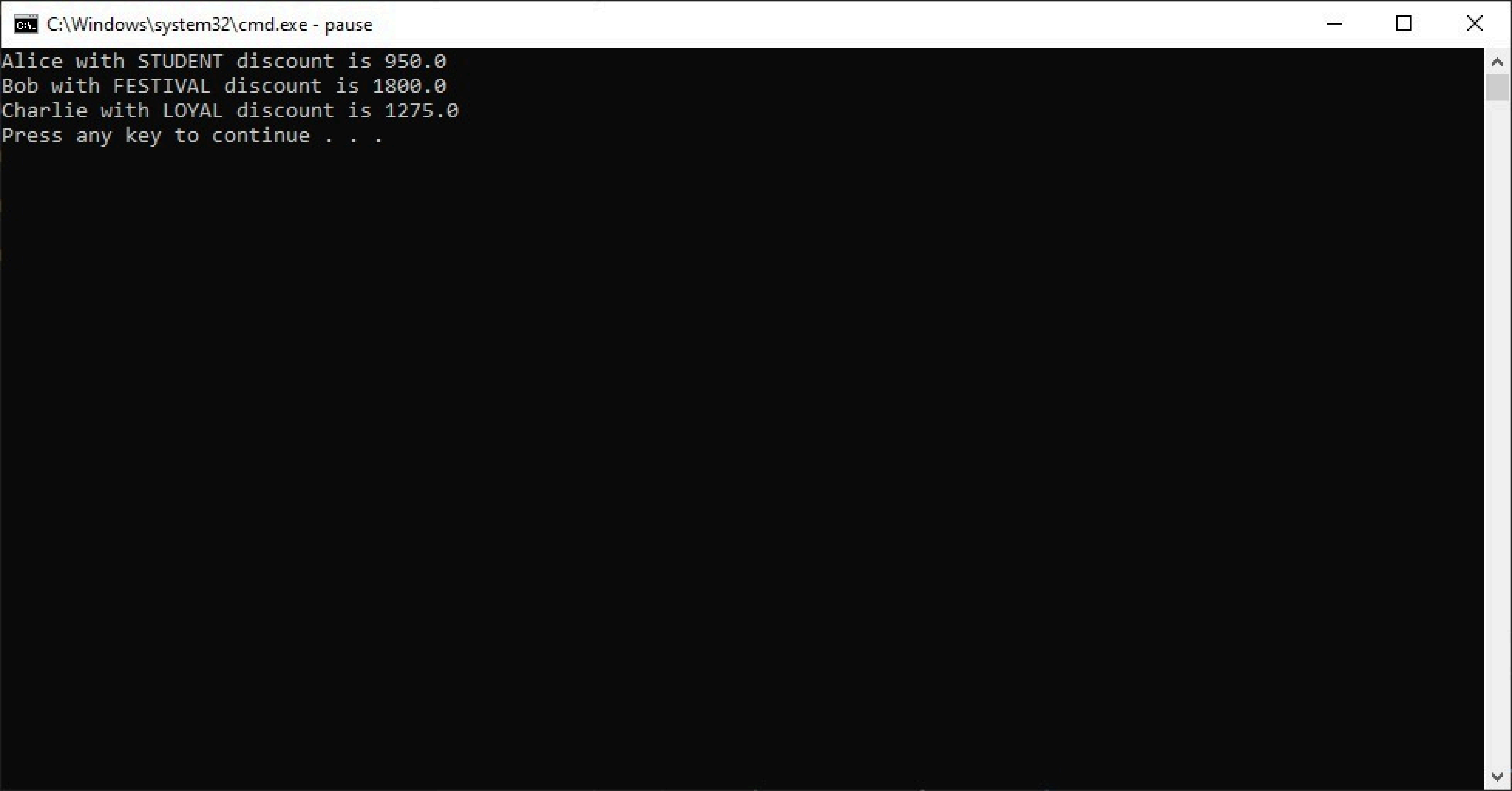
```java
class Student {
    private String name;
    private int marks;
    private String email;

    public Student(String name, int marks, String email) {
        this.name = name;
        this.marks = marks;
        this.email = email;
    }

    // Getters and setters
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getMarks() {
        return marks;
    }

    public void setMarks(int marks) {
        this.marks = marks;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}

class GradeCalculator {
    private double score, total, grade;

    public GradeCalculator(double score, double total) {
        this.score = score;
        this.total = total;
        grade = Calculate();
    }

    private double Calculate() {
        return score * 100 / total;
    }

    public void getGrade() {
        System.out.println(grade);
    }
}

class StudentRepository {
    private String Url;
    private String username;
    private String fileName;

    public StudentRepository(String Url, String username, String fileName) {
        this.Url = Url;
        this.username = username;
        this.fileName = fileName;
    }

    public void saveFileToRepository() {
        System.out.println(username + " Saved " + fileName + " to " + Url);
    }

    public void deleteFileFromRepository() {
        System.out.println(username + " deleted " + fileName + " from " + Url);
    }
}

class EmailService {
    private String service;
    private String reciverEmail;
    private String location;

    public EmailService(String service, String reciverEmail, String location) {
        this.service = service;
        this.reciverEmail = reciverEmail;
        this.location = location;
    }

    public void sendEmail(Student student) {
        System.out.println(student.getEmail() + "is sending email to " + reciverEmail + " using " + service);
    }

    public void serviceLocation () {
        System.out.println("Service location: " + location);
    }
}

public class Lab8Ex1 {
    public static void main(String[] args) {
        Student student = new Student("Alice", 85, "alice@example.com");

        System.out.println("=".repeat(20));

        GradeCalculator gradeCalculator = new GradeCalculator(student.getMarks(), 100);
        System.out.print("Grade: ");
        gradeCalculator.getGrade();

        StudentRepository repository = new StudentRepository("Github.com/wsdq/PDI", "AliceUser", "AliceFile.txt");
        repository.saveFileToRepository();
        repository.deleteFileFromRepository();

        EmailService emailService = new EmailService("Gmail", "bob@example.com", "US Server");
        emailService.sendEmail(student);
        emailService.serviceLocation();

        System.out.println("=".repeat(20));
    }

}
```



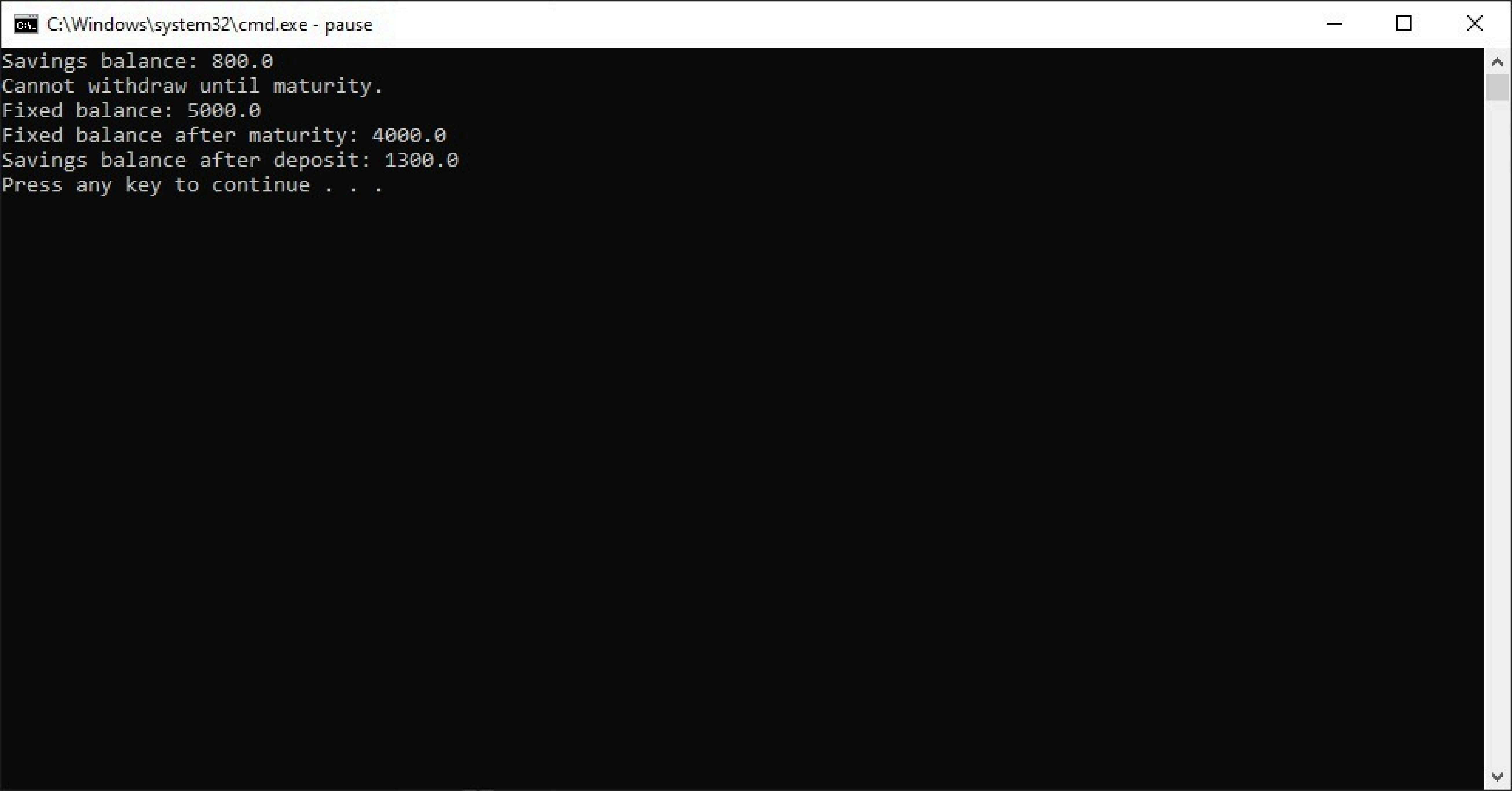
```
====================
Grade: 85.0
AliceUser Saved AliceFile.txt to Github.com/wsdq/PDI
AliceUser deleted AliceFile.txt from Github.com/wsdq/PDI
alice@example.comis sending email to bob@example.com using Gmail
Service location: US Server
====================
Press any key to continue . . .
```

```java
interface calculateDiscount {
    double CalculateDiscount();
}

class Student implements calculateDiscount {
    public String name;
    public double price;
    public String discountType = "STUDENT";

    public Student(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public double CalculateDiscount() {
        return price * 0.95;
    }

    public void displayPrice() {
        System.out.println(name + " with " + discountType + " discount is " + CalculateDiscount());
    }
}

class FESTIVAL implements calculateDiscount {
    public String name;
    public double price;
    public String discountType = "FESTIVAL";

    public FESTIVAL(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public double CalculateDiscount() {
        return price * 0.9;
    }

    public void displayPrice() {
        System.out.println(name + " with " + discountType + " discount is " + CalculateDiscount());
    }
}

class LOYAL implements calculateDiscount {
    public String name;
    public double price;
    public String discountType = "LOYAL";

    public LOYAL(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public double CalculateDiscount() {
        return price * 0.85;
    }

    public void displayPrice() {
        System.out.println(name + " with " + discountType + " discount is " + CalculateDiscount());
    }
}

public class Lab8Ex2 {
    public static void main(String[] args) {
        Student studentDiscount = new Student("Alice", 1000);
        FESTIVAL festivalDiscount = new FESTIVAL("Bob", 2000);
        LOYAL loyalDiscount = new LOYAL("Charlie", 1500);

        studentDiscount.displayPrice();
        festivalDiscount.displayPrice();
        loyalDiscount.displayPrice();
    }
}
```



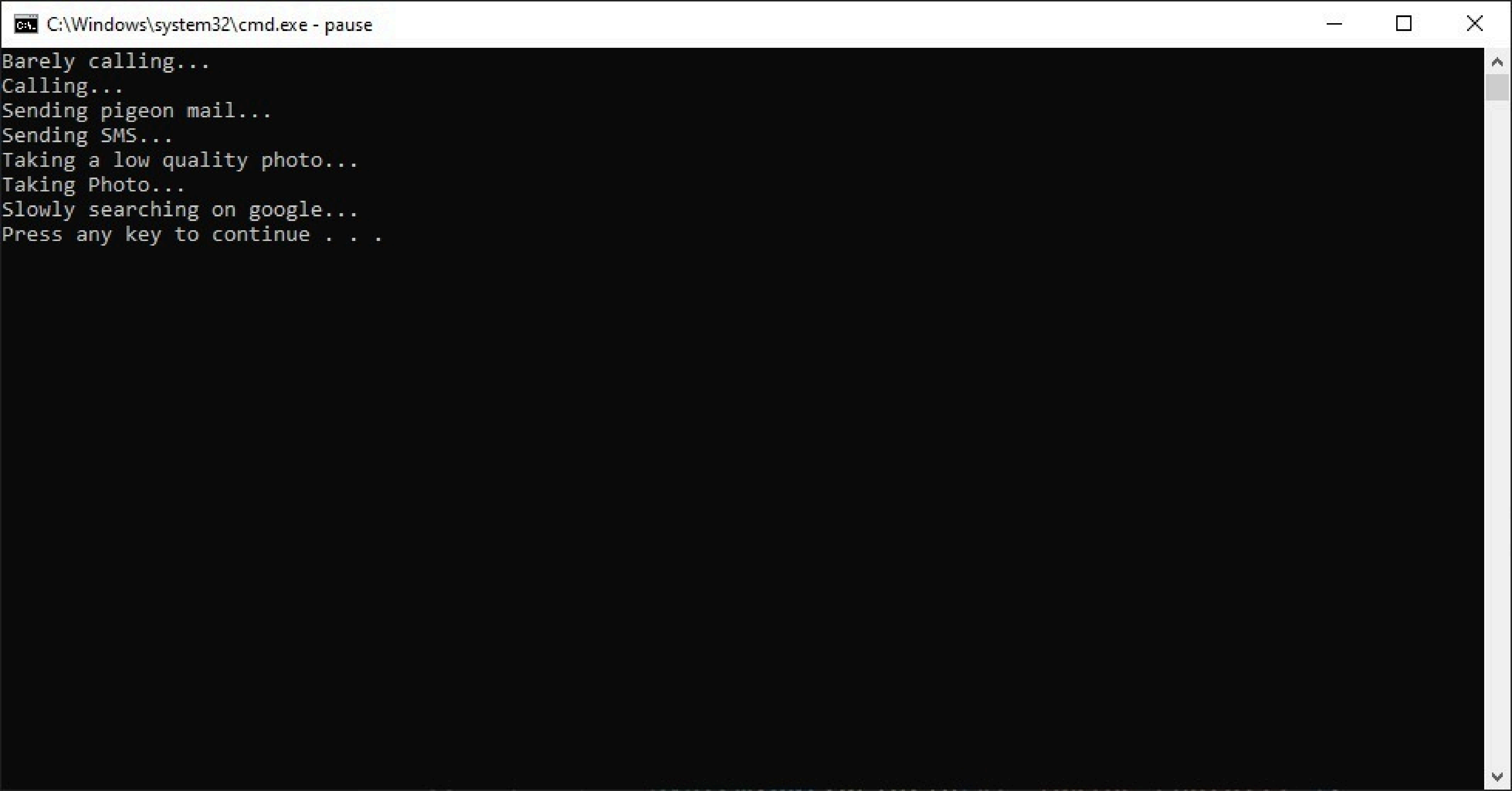
```
Alice with STUDENT discount is 950.0
Bob with FESTIVAL discount is 1800.0
Charlie with LOYAL discount is 1275.0
Press any key to continue . . .
```

```java
abstract class Account {
    protected String accountNumber;
    protected double balance;
    protected String accountType;

    public Account(String accountNumber, double balance, String accountType) {
        this.accountNumber = accountNumber;
        this.balance = balance;
        this.accountType = accountType;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
        }
    }

    public double getBalance() {
        return balance;
    }

    public abstract void withdraw(double amount);
}

class SavingsAccount extends Account {

    public SavingsAccount(String accountNumber, double balance) {
        super(accountNumber, balance, "Savings");
    }

    @Override
    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
        } else {
            System.out.println("Insufficient balance or invalid amount.");
        }
    }
}

class FixedDepositAccount extends Account {
    private boolean matured;

    public FixedDepositAccount(String accountNumber, double balance, boolean matured) {
        super(accountNumber, balance, "Fixed Deposit");
        this.matured = matured;
    }

    @Override
    public void withdraw(double amount) {
        if (matured) {
            if (amount > 0 && amount <= balance) {
                balance -= amount;
            } else {
                System.out.println("Invalid amount or insufficient balance.");
            }
        } else {
            System.out.println("Cannot withdraw until maturity.");
        }
    }

    public void setMatured(boolean matured) {
        this.matured = matured;
    }
}

public class Lab8Ex3 {
    public static void main(String[] args) {

        Account savings = new SavingsAccount("S001", 1000);
        Account fixed = new FixedDepositAccount("F001", 5000, false);

        savings.withdraw(200);
        System.out.println("Savings balance: " + savings.getBalance());

        fixed.withdraw(1000); // restricted
        System.out.println("Fixed balance: " + fixed.getBalance());

        ((FixedDepositAccount) fixed).setMatured(true);
        fixed.withdraw(1000); // now allowed
        System.out.println("Fixed balance after maturity: " + fixed.getBalance());

        // Deposit example
        savings.deposit(500);
        System.out.println("Savings balance after deposit: " + savings.getBalance());
    }
}

```



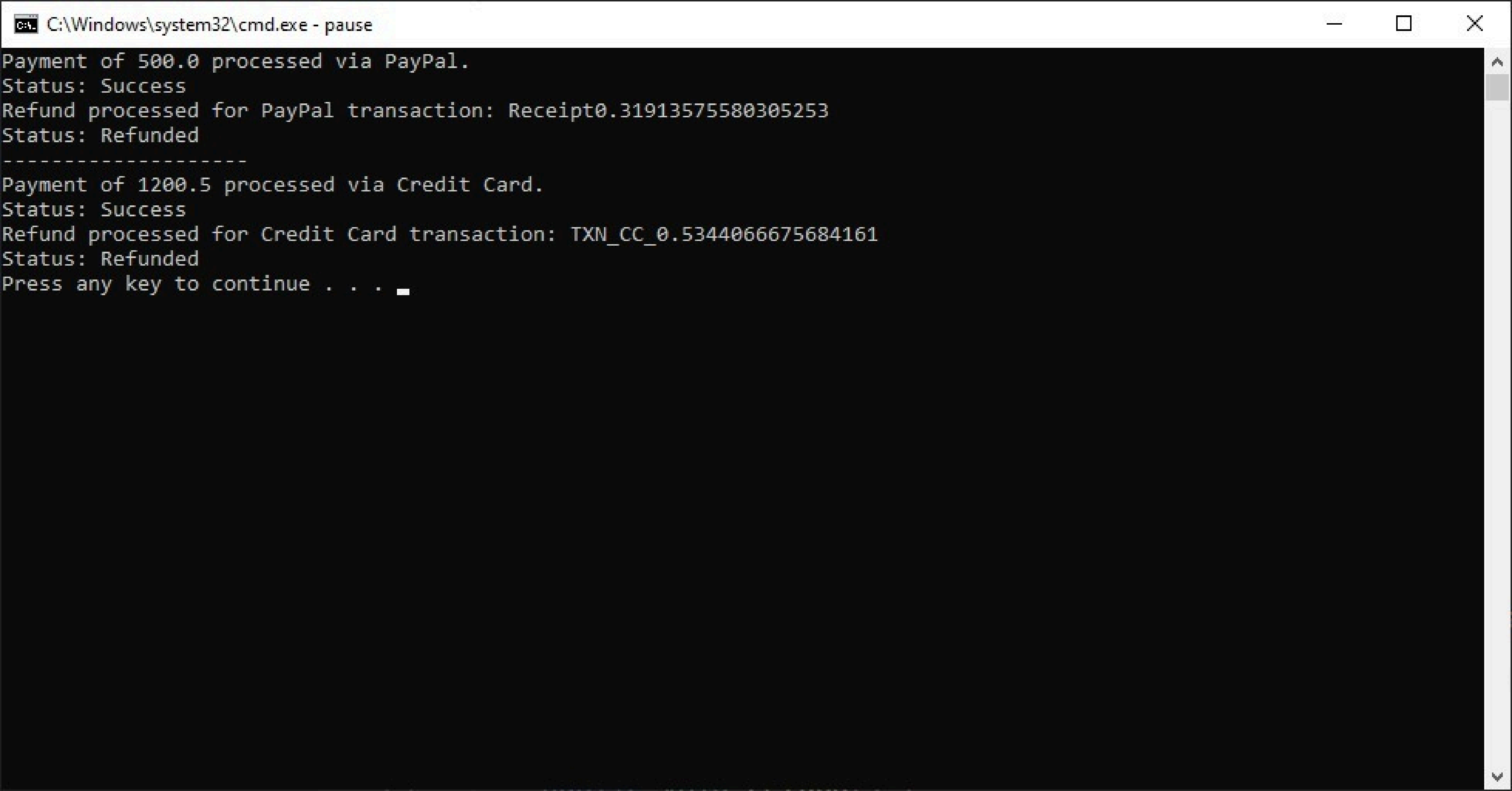
```
Savings balance: 800.0
Cannot withdraw until maturity.
Fixed balance: 5000.0
Fixed balance after maturity: 4000.0
Savings balance after deposit: 1300.0
Press any key to continue . . .
```

```java
interface CallDevice {
    void call();
}

interface SMSDevice {
    void sendSMS();
}

interface InternetDevice {
    void browseInternet();
}

interface CameraDevice {
    void takePhoto();
}

class BasicPhone implements CallDevice, SMSDevice, CameraDevice {
    public int battery;
    public int service;
    public final String version = "Nokia";

    public BasicPhone(int battery, int service) {
        this.battery = battery;
        this.service = service;
    }

    public void call() {
        System.out.println("Barely calling...");
    }

    public void sendSMS() {
        System.out.println("Sending pigeon mail...");
    }

    public void takePhoto() {
        System.out.println("Taking a low quality photo...");
    }
}

class SmartPhone implements CallDevice, SMSDevice, CameraDevice, InternetDevice {
    public int battery;
    public int service;
    public final String version = "Red Magic";

    public SmartPhone(int battery, int service) {
        this.battery = battery;
        this.service = service;
    }

    public void call() {
        System.out.println("Calling...");
    }

    public void sendSMS() {
        System.out.println("Sending SMS...");
    }

    public void takePhoto() {
        System.out.println("Taking Photo...");
    }

    public void browseInternet() {
        if (service > 50) {
            System.out.println("Searching on google...");
        } else {
            System.out.println("Slowly searching on google...");
        }
    }
}

public class Lab8Ex4 {
    public static void main(String[] args) {
        BasicPhone basicPhone = new BasicPhone(100, 100);
        SmartPhone smartPhone = new SmartPhone(90, 50);

        basicPhone.call();
        smartPhone.call();
        basicPhone.sendSMS();
        smartPhone.sendSMS();
        basicPhone.takePhoto();
        smartPhone.takePhoto();

        smartPhone.browseInternet();

    }
}
```



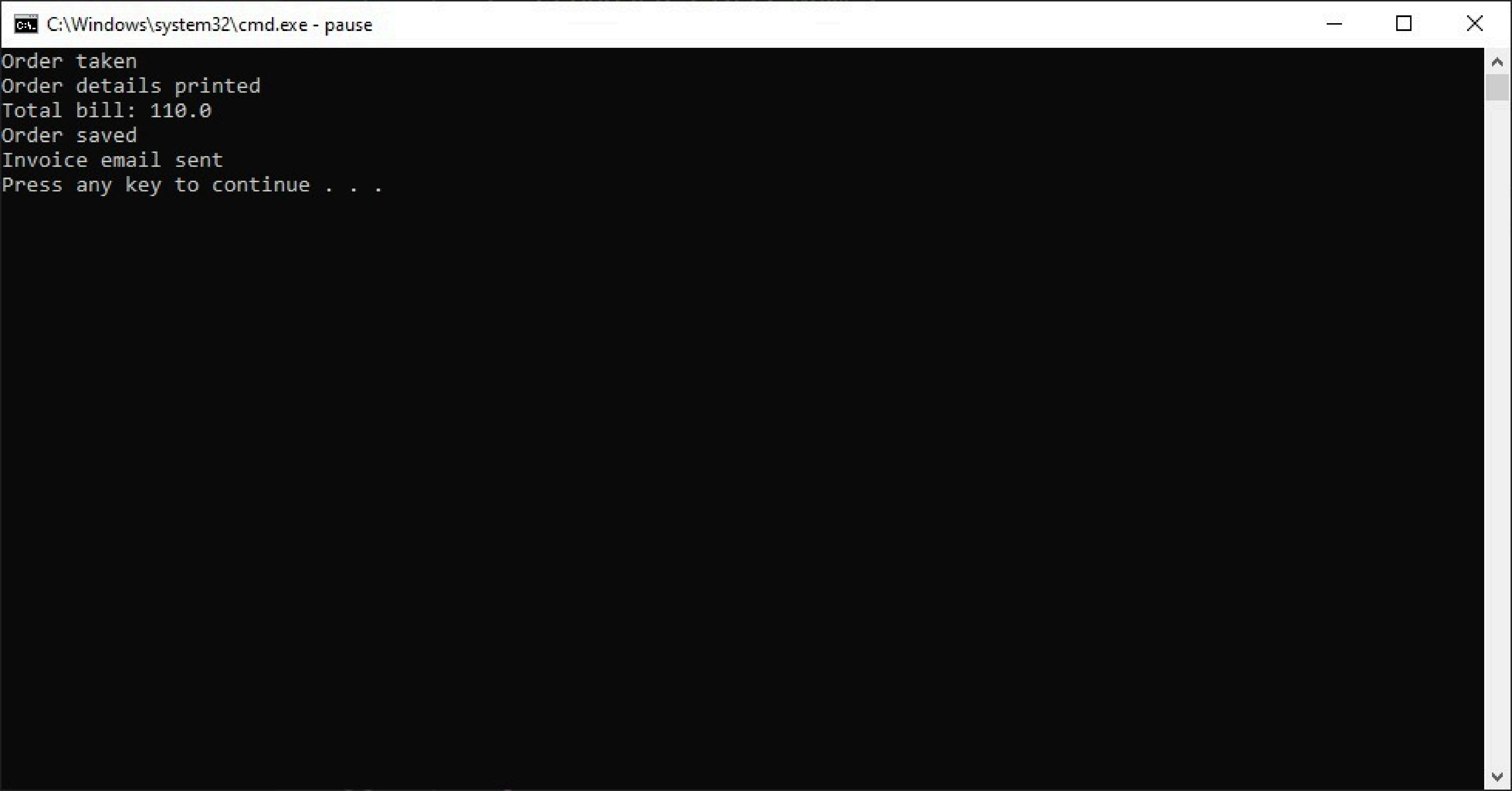
```
Barely calling...
Calling...
Sending pigeon mail...
Sending SMS...
Taking a low quality photo...
Taking Photo...
Slowly searching on google...
Press any key to continue . . .
```

```java
interface Payment {
    void pay(double amount);

    void refund();

    String getStatus();
}

class PayPal implements Payment {
    private String transactionId;
    private double amount;
    private String status;

    @Override
    public void pay(double amount) {
        this.amount = amount;
        this.transactionId = "Receipt" + Math.random();
        this.status = "Success";
        System.out.println("Payment of " + amount + " processed via PayPal.");
    }

    @Override
    public void refund() {
        this.status = "Refunded";
        System.out.println("Refund processed for PayPal transaction: " + transactionId);
    }

    @Override
    public String getStatus() {
        return this.status;
    }
}

class CreditCard implements Payment {
    private String transactionId;
    private double amount;
    private String status;

    @Override
    public void pay(double amount) {
        this.amount = amount;
        this.transactionId = "TXN_CC_" + Math.random();
        this.status = "Success";
        System.out.println("Payment of " + amount + " processed via Credit Card.");
    }

    @Override
    public void refund() {
        this.status = "Refunded";
        System.out.println("Refund processed for Credit Card transaction: " + transactionId);
    }

    @Override
    public String getStatus() {
        return this.status;
    }
}

class ShoppingCart {
    private Payment payment;

    public ShoppingCart(Payment payment) {
        this.payment = payment;
    }

    public void checkout(double amount) {
        payment.pay(amount);
    }
}

public class Lab8Ex5 {
    public static void main(String[] args) {
        Payment paypal = new PayPal();
        ShoppingCart cart1 = new ShoppingCart(paypal);
        cart1.checkout(500.0);
        System.out.println("Status: " + paypal.getStatus());
        paypal.refund();
        System.out.println("Status: " + paypal.getStatus());

        System.out.println("--------------------");

        Payment creditCard = new CreditCard();
        ShoppingCart cart2 = new ShoppingCart(creditCard);
        cart2.checkout(1200.50);
        System.out.println("Status: " + creditCard.getStatus());
        creditCard.refund();
        System.out.println("Status: " + creditCard.getStatus());
    }
}
```



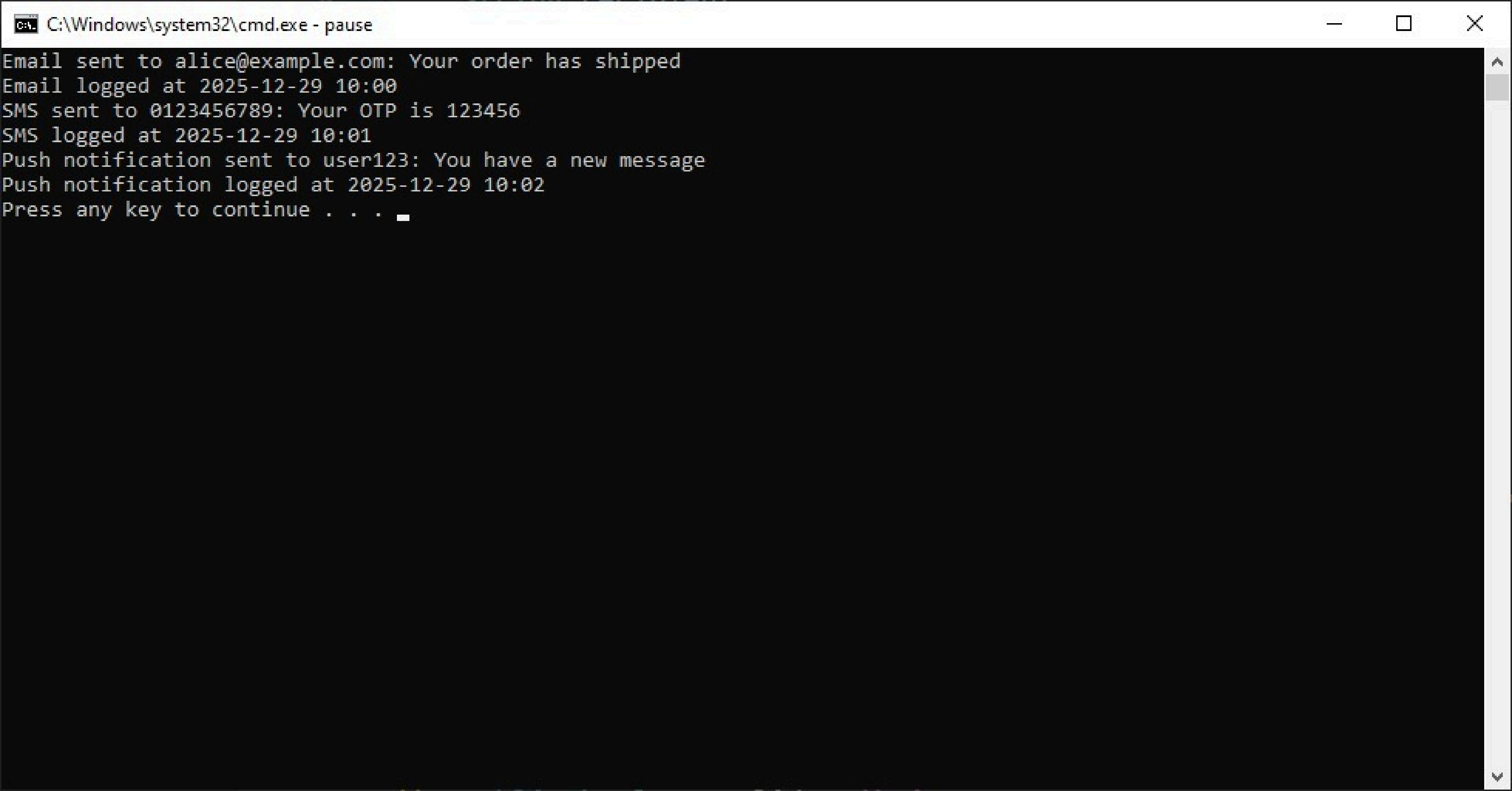
```
Payment of 500.0 processed via PayPal.
Status: Success
Refund processed for PayPal transaction: Receipt0.31913575580305253
Status: Refunded
--------------------
Payment of 1200.5 processed via Credit Card.
Status: Success
Refund processed for Credit Card transaction: TXN_CC_0.5344066675684161
Status: Refunded
Press any key to continue . . .
```

```java
/*
 * 1. Violated SOLID Principle:
 *    - Single Responsibility Principle (SRP): The original OrderService class handled
 *      too many distinct responsibilities: taking orders, calculating bills, saving orders,
 *      and sending emails. Each of these responsibilities has now been moved to a separate class.
 *
 * 2. Proposed Class Names:
 *    - OrderManager (Handles order operations: take, update, print)
 *    - BillingService (Handles bill calculation: apply tax, calculate, print)
 *    - OrderRepository (Handles database operations: save, delete, find)
 *    - EmailService (Handles notifications: send invoice, confirmation, reminder)
 */


class OrderManager {
    int orderId;
    String customerName;
    int quantity;

    void takeOrder() {
        System.out.println("Order taken");
    }

    void updateOrder() {
        System.out.println("Order updated");
    }

    void printOrder() {
        System.out.println("Order details printed");
    }
}

// BillingService.java
class BillingService {
    double price;
    double tax;
    double totalAmount;

    void calculateBill() {
        totalAmount = price + tax;
    }

    void applyTax() {
        tax = price * 0.10;
    }

    void printBill() {
        System.out.println("Total bill: " + totalAmount);
    }
}

// OrderRepository.java
class OrderRepository {
    int orderId;
    String databaseName;
    boolean isSaved;

    void saveOrder() {
        isSaved = true;
        System.out.println("Order saved");
    }

    void deleteOrder() {
        System.out.println("Order deleted");
    }

    void findOrder() {
        System.out.println("Order found");
    }
}

// EmailService.java
class EmailService {
    String senderEmail;
    String receiverEmail;
    String message;

    void sendInvoiceEmail() {
        System.out.println("Invoice email sent");
    }

    void sendConfirmation() {
        System.out.println("Confirmation email sent");
    }

    void sendReminder() {
        System.out.println("Reminder email sent");
    }
}

public class Lab8Ex6 {
    public static void main(String[] args) {

        OrderManager order = new OrderManager();
        order.orderId = 101;
        order.customerName = "Alice";
        order.quantity = 3;
        order.takeOrder();
        order.printOrder();

        BillingService billing = new BillingService();
        billing.price = 100;
        billing.applyTax();
        billing.calculateBill();
        billing.printBill();

        OrderRepository repo = new OrderRepository();
        repo.orderId = order.orderId;
        repo.databaseName = "OrdersDB";
        repo.saveOrder();

        EmailService email = new EmailService();
        email.senderEmail = "shop@example.com";
        email.receiverEmail = "alice@example.com";
        email.message = "Your order invoice";
        email.sendInvoiceEmail();
    }
}
```



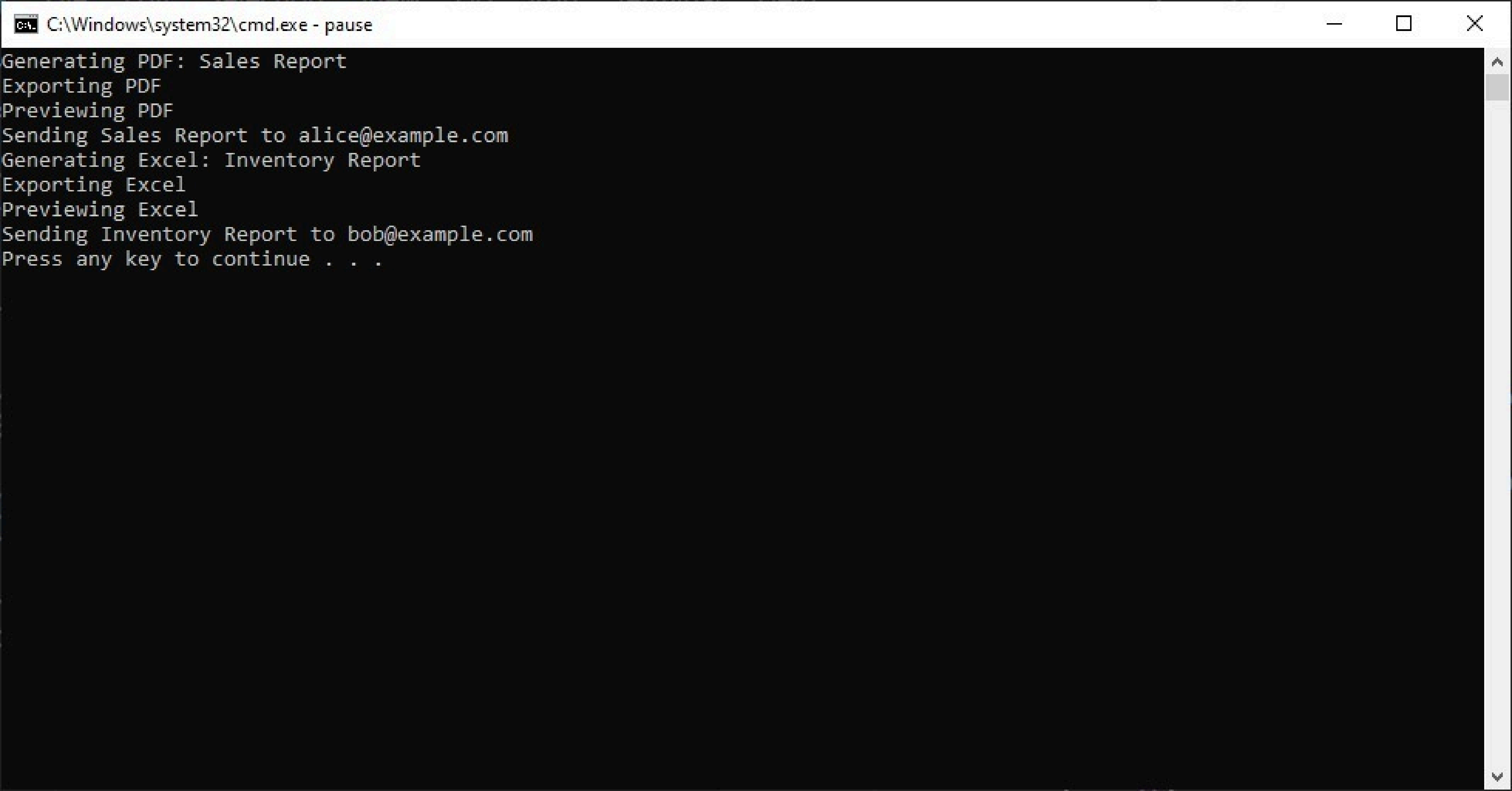
```
Order taken
Order details printed
Total bill: 110.0
Order saved
Invoice email sent
Press any key to continue . . .
```

```java
interface Notification {
    void send();
    boolean validate();
    void logNotification();
}

class EmailNotification implements Notification {
    String recipient;
    String message;
    String timestamp;

    EmailNotification(String recipient, String message) {
        this.recipient = recipient;
        this.message = message;
        this.timestamp = "2025-12-29 10:00";
    }

    public void send() {
        if (validate()) {
            System.out.println("Email sent to " + recipient + ": " + message);
            logNotification();
        } else {
            System.out.println("Invalid email notification");
        }
    }

    public boolean validate() {
        return recipient.contains("@") && !message.isEmpty();
    }

    public void logNotification() {
        System.out.println("Email logged at " + timestamp);
    }
}

class SMSNotification implements Notification {
    String recipient;
    String message;
    String timestamp;

    SMSNotification(String recipient, String message) {
        this.recipient = recipient;
        this.message = message;
        this.timestamp = "2025-12-29 10:01";
    }

    public void send() {
        if (validate()) {
            System.out.println("SMS sent to " + recipient + ": " + message);
            logNotification();
        } else {
            System.out.println("Invalid SMS notification");
        }
    }

    public boolean validate() {
        return recipient.matches("\\d{10}") && !message.isEmpty();
    }

    public void logNotification() {
        System.out.println("SMS logged at " + timestamp);
    }
}

class PushNotification implements Notification {
    String recipient;
    String message;
    String timestamp;

    PushNotification(String recipient, String message) {
        this.recipient = recipient;
        this.message = message;
        this.timestamp = "2025-12-29 10:02";
    }

    public void send() {
        if (validate()) {
            System.out.println("Push notification sent to " + recipient + ": " + message);
            logNotification();
        } else {
            System.out.println("Invalid push notification");
        }
    }

    public boolean validate() {
        return !recipient.isEmpty() && !message.isEmpty();
    }

    public void logNotification() {
        System.out.println("Push notification logged at " + timestamp);
    }
}

class NotificationService {
    void sendNotification(Notification notification) {
        notification.send();
    }
}

public class Lab8Ex7 {
    public static void main(String[] args) {
        NotificationService service = new NotificationService();

        Notification email = new EmailNotification("alice@example.com", "Your order has shipped");
        Notification sms = new SMSNotification("0123456789", "Your OTP is 123456");
        Notification push = new PushNotification("user123", "You have a new message");

        service.sendNotification(email);
        service.sendNotification(sms);
        service.sendNotification(push);
    }
}
```


C:\Windows\system32\cmd.exe - pause

```
Email sent to alice@example.com: Your order has shipped
Email logged at 2025-12-29 10:00
SMS sent to 0123456789: Your OTP is 123456
SMS logged at 2025-12-29 10:01
Push notification sent to user123: You have a new message
Push notification logged at 2025-12-29 10:02
Press any key to continue . . .
```

```java
class Report {
    String reportName;
    String createdDate;
    String format;

    Report(String reportName, String createdDate, String format) {
        this.reportName = reportName;
        this.createdDate = createdDate;
        this.format = format;
    }
}

interface ReportGenerator {
    void generate();
    void export();
    void preview();
}

class PDFReportGenerator implements ReportGenerator {
    Report report;

    PDFReportGenerator(Report report) {
        this.report = report;
    }

    public void generate() {
        System.out.println("Generating PDF: " + report.reportName);
    }

    public void export() {
        System.out.println("Exporting PDF");
    }

    public void preview() {
        System.out.println("Previewing PDF");
    }
}

class ExcelReportGenerator implements ReportGenerator {
    Report report;

    ExcelReportGenerator(Report report) {
        this.report = report;
    }

    public void generate() {
        System.out.println("Generating Excel: " + report.reportName);
    }

    public void export() {
        System.out.println("Exporting Excel");
    }

    public void preview() {
        System.out.println("Previewing Excel");
    }
}

interface ReportDelivery {
    void send();
}

class EmailDelivery implements ReportDelivery {
    Report report;
    String recipient;

    EmailDelivery(Report report, String recipient) {
        this.report = report;
        this.recipient = recipient;
    }

    public void send() {
        System.out.println("Sending " + report.reportName + " to " + recipient);
    }
}

public class Lab8Ex8 {
    public static void main(String[] args) {
        Report pdfReport = new Report("Sales Report", "2025-12-29", "PDF");
        ReportGenerator pdfGen = new PDFReportGenerator(pdfReport);
        ReportDelivery emailPdf = new EmailDelivery(pdfReport, "alice@example.com");

        pdfGen.generate();
        pdfGen.export();
        pdfGen.preview();
        emailPdf.send();

        Report excelReport = new Report("Inventory Report", "2025-12-29", "Excel");
        ReportGenerator excelGen = new ExcelReportGenerator(excelReport);
        ReportDelivery emailExcel = new EmailDelivery(excelReport, "bob@example.com");

        excelGen.generate();
        excelGen.export();
        excelGen.preview();
        emailExcel.send();
    }
}
```



```
Generating PDF: Sales Report
Exporting PDF
Previewing PDF
Sending Sales Report to alice@example.com
Generating Excel: Inventory Report
Exporting Excel
Previewing Excel
Sending Inventory Report to bob@example.com
Press any key to continue . . .
```