

```
1  class Student {
2      private String name;
3      private int marks;
4      private String email;
5
6      public Student(String name, int marks, String email) {
7          this.name = name;
8          this.marks = marks;
9          this.email = email;
10     }
11
12     // Getters and setters
13     public String getName() {
14         return name;
15     }
16
17     public void setName(String name) {
18         this.name = name;
19     }
20
21     public int getMarks() {
22         return marks;
23     }
24
25     public void setMarks(int marks) {
26         this.marks = marks;
27     }
28
29     public String getEmail() {
30         return email;
31     }
32
33     public void setEmail(String email) {
34         this.email = email;
35     }
36 }
37
38 class GradeCalculator {
39     private double score, total, grade;
40
41     public GradeCalculator(double score, double total) {
42         this.score = score;
43         this.total = total;
44         grade = Calculate();
45     }
46
47     private double Calculate() {
48         return score * 100 / total;
49     }
50
51     public void getGrade() {
52         System.out.println(grade);
53     }
54 }
55
56 class StudentRepository {
57     private String Url;
58     private String username;
59     private String fileName;
60
61     public StudentRepository(String Url, String username, String fileName) {
62         this.Url = Url;
63         this.username = username;
64         this.fileName = fileName;
65     }
66
67     public void saveFileToRepository() {
68         System.out.println(username + " Saved " + fileName + " to " + Url);
69     }
70
71     public void deleteFileFromRepository() {
72         System.out.println(username + " deleted " + fileName + " from " + Url);
73     }
74 }
75
76 class EmailService {
77     private String service;
78     private String reciverEmail;
79     private String location;
80
81     public EmailService(String service, String reciverEmail, String location) {
82         this.service = service;
83         this.reciverEmail = reciverEmail;
84         this.location = location;
85     }
86
87     public void sendEmail(Student student) {
88         System.out.println(student.getEmail() + "is sending email to " + reciverEmail + " using " + service);
89     }
90
91     public void serviceLocation () {
92         System.out.println("Service location: " + location);
93     }
94 }
95
96 public class Lab8Ex1 {
97     public static void main(String[] args) {
98         Student student = new Student("Alice", 85, "alice@example.com");
99
100        System.out.println("=".repeat(20));
101
102        GradeCalculator gradeCalculator = new GradeCalculator(student.getMarks(), 100);
103        System.out.print("Grade: ");
104        gradeCalculator.getGrade();
105
106        StudentRepository repository = new StudentRepository("Github.com/wsdq/PDI", "AliceUser", "AliceFile.txt");
107        repository.saveFileToRepository();
108        repository.deleteFileFromRepository();
109
110        EmailService emailService = new EmailService("Gmail", "bob@example.com", "US Server");
111        emailService.sendEmail(student);
112        emailService.serviceLocation();
113
114        System.out.println("=".repeat(20));
115    }
116 }
117 }
118
119
```



```
C:\Windows\system32\cmd.exe - pause
=====
Grade: 85.0
AliceUser Saved AliceFile.txt to Github.com/wsdq/PDI
AliceUser deleted AliceFile.txt from Github.com/wsdq/PDI
alice@example.com is sending email to bob@example.com using Gmail
Service location: US Server
=====
Press any key to continue . . .
```



```
1 interface calculateDiscount {
2     double CalculateDiscount();
3 }
4
5 class Student implements calculateDiscount {
6     public String name;
7     public double price;
8     public String discountType = "STUDENT";
9
10    public Student(String name, double price) {
11        this.name = name;
12        this.price = price;
13    }
14
15    public double CalculateDiscount() {
16        return price * 0.95;
17    }
18
19    public void displayPrice() {
20        System.out.println(name + " with " + discountType + " discount is " + CalculateDiscount());
21    }
22 }
23
24 class FESTIVAL implements calculateDiscount {
25     public String name;
26     public double price;
27     public String discountType = "FESTIVAL";
28
29    public FESTIVAL(String name, double price) {
30        this.name = name;
31        this.price = price;
32    }
33
34    public double CalculateDiscount() {
35        return price * 0.9;
36    }
37
38    public void displayPrice() {
39        System.out.println(name + " with " + discountType + " discount is " + CalculateDiscount());
40    }
41 }
42
43 class LOYAL implements calculateDiscount {
44     public String name;
45     public double price;
46     public String discountType = "LOYAL";
47
48    public LOYAL(String name, double price) {
49        this.name = name;
50        this.price = price;
51    }
52
53    public double CalculateDiscount() {
54        return price * 0.85;
55    }
56
57    public void displayPrice() {
58        System.out.println(name + " with " + discountType + " discount is " + CalculateDiscount());
59    }
60 }
61
62 public class Lab8Ex2 {
63     public static void main(String[] args) {
64         Student studentDiscount = new Student("Alice", 1000);
65         FESTIVAL festivalDiscount = new FESTIVAL("Bob", 2000);
66         LOYAL loyalDiscount = new LOYAL("Charlie", 1500);
67
68         studentDiscount.displayPrice();
69         festivalDiscount.displayPrice();
70         loyalDiscount.displayPrice();
71     }
72 }
73
```



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



```
1  abstract class Account {
2      protected String accountNumber;
3      protected double balance;
4      protected String accountType;
5
6      public Account(String accountNumber, double balance, String accountType) {
7          this.accountNumber = accountNumber;
8          this.balance = balance;
9          this.accountType = accountType;
10     }
11
12     public void deposit(double amount) {
13         if (amount > 0) {
14             balance += amount;
15         }
16     }
17
18     public double getBalance() {
19         return balance;
20     }
21
22     public abstract void withdraw(double amount);
23 }
24
25 class SavingsAccount extends Account {
26
27     public SavingsAccount(String accountNumber, double balance) {
28         super(accountNumber, balance, "Savings");
29     }
30
31     @Override
32     public void withdraw(double amount) {
33         if (amount > 0 && amount <= balance) {
34             balance -= amount;
35         } else {
36             System.out.println("Insufficient balance or invalid amount.");
37         }
38     }
39 }
40
41 class FixedDepositAccount extends Account {
42     private boolean matured;
43
44     public FixedDepositAccount(String accountNumber, double balance, boolean matured) {
45         super(accountNumber, balance, "Fixed Deposit");
46         this.matured = matured;
47     }
48
49     @Override
50     public void withdraw(double amount) {
51         if (matured) {
52             if (amount > 0 && amount <= balance) {
53                 balance -= amount;
54             } else {
55                 System.out.println("Invalid amount or insufficient balance.");
56             }
57         } else {
58             System.out.println("Cannot withdraw until maturity.");
59         }
60     }
61
62     public void setMatured(boolean matured) {
63         this.matured = matured;
64     }
65 }
66
67 public class Lab8Ex3 {
68     public static void main(String[] args) {
69
70         Account savings = new SavingsAccount("S001", 1000);
71         Account fixed = new FixedDepositAccount("F001", 5000, false);
72
73         savings.withdraw(200);
74         System.out.println("Savings balance: " + savings.getBalance());
75
76         fixed.withdraw(1000); // restricted
77         System.out.println("Fixed balance: " + fixed.getBalance());
78
79         ((FixedDepositAccount) fixed).setMatured(true);
80         fixed.withdraw(1000); // now allowed
81         System.out.println("Fixed balance after maturity: " + fixed.getBalance());
82
83         // Deposit example
84         savings.deposit(500);
85         System.out.println("Savings balance after deposit: " + savings.getBalance());
86     }
87 }
88
```



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



```
1  interface CallDevice {
2      void call();
3  }
4
5  interface SMSDevice {
6      void sendSMS();
7  }
8
9  interface InternetDevice {
10     void browseInternet();
11 }
12
13 interface CameraDevice {
14     void takePhoto();
15 }
16
17 class BasicPhone implements CallDevice, SMSDevice, CameraDevice {
18     public int battery;
19     public int service;
20     public final String version = "Nokia";
21
22     public BasicPhone(int battery, int service) {
23         this.battery = battery;
24         this.service = service;
25     }
26
27     public void call() {
28         System.out.println("Barely calling...");
29     }
30
31     public void sendSMS() {
32         System.out.println("Sending pigeon mail...");
33     }
34
35     public void takePhoto() {
36         System.out.println("Taking a low quality photo...");
37     }
38 }
39
40 class SmartPhone implements CallDevice, SMSDevice, CameraDevice, InternetDevice {
41     public int battery;
42     public int service;
43     public final String version = "Red Magic";
44
45     public SmartPhone(int battery, int service) {
46         this.battery = battery;
47         this.service = service;
48     }
49
50     public void call() {
51         System.out.println("Calling...");
52     }
53
54     public void sendSMS() {
55         System.out.println("Sending SMS...");
56     }
57
58     public void takePhoto() {
59         System.out.println("Taking Photo...");
60     }
61
62     public void browseInternet() {
63         if (service > 50) {
64             System.out.println("Searching on google...");
65         } else {
66             System.out.println("Slowly searching on google...");
67         }
68     }
69 }
70
71 public class Lab8Ex4 {
72     public static void main(String[] args) {
73         BasicPhone basicPhone = new BasicPhone(100, 100);
74         SmartPhone smartPhone = new SmartPhone(90, 50);
75
76         basicPhone.call();
77         smartPhone.call();
78         basicPhone.sendSMS();
79         smartPhone.sendSMS();
80         basicPhone.takePhoto();
81         smartPhone.takePhoto();
82
83         smartPhone.browseInternet();
84
85     }
86 }
```


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


```
1 interface Payment {
2     void pay(double amount);
3
4     void refund();
5
6     String getStatus();
7 }
8
9 class PayPal implements Payment {
10     private String transactionId;
11     private double amount;
12     private String status;
13
14     @Override
15     public void pay(double amount) {
16         this.amount = amount;
17         this.transactionId = "Receipt" + Math.random();
18         this.status = "Success";
19         System.out.println("Payment of " + amount + " processed via PayPal.");
20     }
21
22     @Override
23     public void refund() {
24         this.status = "Refunded";
25         System.out.println("Refund processed for PayPal transaction: " + transactionId);
26     }
27
28     @Override
29     public String getStatus() {
30         return this.status;
31     }
32 }
33
34 class CreditCard implements Payment {
35     private String transactionId;
36     private double amount;
37     private String status;
38
39     @Override
40     public void pay(double amount) {
41         this.amount = amount;
42         this.transactionId = "TXN_CC_" + Math.random();
43         this.status = "Success";
44         System.out.println("Payment of " + amount + " processed via Credit Card.");
45     }
46
47     @Override
48     public void refund() {
49         this.status = "Refunded";
50         System.out.println("Refund processed for Credit Card transaction: " + transactionId);
51     }
52
53     @Override
54     public String getStatus() {
55         return this.status;
56     }
57 }
58
59 class ShoppingCart {
60     private Payment payment;
61
62     public ShoppingCart(Payment payment) {
63         this.payment = payment;
64     }
65
66     public void checkout(double amount) {
67         payment.pay(amount);
68     }
69 }
70
71 public class Lab8Ex5 {
72     public static void main(String[] args) {
73         Payment paypal = new PayPal();
74         ShoppingCart cart1 = new ShoppingCart(paypal);
75         cart1.checkout(500.0);
76         System.out.println("Status: " + paypal.getStatus());
77         paypal.refund();
78         System.out.println("Status: " + paypal.getStatus());
79
80         System.out.println("-----");
81
82         Payment creditCard = new CreditCard();
83         ShoppingCart cart2 = new ShoppingCart(creditCard);
84         cart2.checkout(1200.50);
85         System.out.println("Status: " + creditCard.getStatus());
86         creditCard.refund();
87         System.out.println("Status: " + creditCard.getStatus());
88     }
89 }
```



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



```
1  /*
2  * 1. Violated SOLID Principle:
3  *   - Single Responsibility Principle (SRP): The original OrderService class handled
4  *     too many distinct responsibilities: taking orders, calculating bills, saving orders,
5  *     and sending emails. Each of these responsibilities has now been moved to a separate class.
6  *
7  * 2. Proposed Class Names:
8  *   - OrderManager (Handles order operations: take, update, print)
9  *   - BillingService (Handles bill calculation: apply tax, calculate, print)
10 *   - OrderRepository (Handles database operations: save, delete, find)
11 *   - EmailService (Handles notifications: send invoice, confirmation, reminder)
12 */
13
14
15 class OrderManager {
16     int orderId;
17     String customerName;
18     int quantity;
19
20     void takeOrder() {
21         System.out.println("Order taken");
22     }
23
24     void updateOrder() {
25         System.out.println("Order updated");
26     }
27
28     void printOrder() {
29         System.out.println("Order details printed");
30     }
31 }
32
33 // BillingService.java
34 class BillingService {
35     double price;
36     double tax;
37     double totalAmount;
38
39     void calculateBill() {
40         totalAmount = price + tax;
41     }
42
43     void applyTax() {
44         tax = price * 0.10;
45     }
46
47     void printBill() {
48         System.out.println("Total bill: " + totalAmount);
49     }
50 }
51
52 // OrderRepository.java
53 class OrderRepository {
54     int orderId;
55     String databaseName;
56     boolean isSaved;
57
58     void saveOrder() {
59         isSaved = true;
60         System.out.println("Order saved");
61     }
62
63     void deleteOrder() {
64         System.out.println("Order deleted");
65     }
66
67     void findOrder() {
68         System.out.println("Order found");
69     }
70 }
71
72 // EmailService.java
73 class EmailService {
74     String senderEmail;
75     String receiverEmail;
76     String message;
77
78     void sendInvoiceEmail() {
79         System.out.println("Invoice email sent");
80     }
81
82     void sendConfirmation() {
83         System.out.println("Confirmation email sent");
84     }
85
86     void sendReminder() {
87         System.out.println("Reminder email sent");
88     }
89 }
90
91 public class Lab8Ex6 {
92     public static void main(String[] args) {
93
94         OrderManager order = new OrderManager();
95         order.orderId = 101;
96         order.customerName = "Alice";
97         order.quantity = 3;
98         order.takeOrder();
99         order.printOrder();
100
101         BillingService billing = new BillingService();
102         billing.price = 100;
103         billing.applyTax();
104         billing.calculateBill();
105         billing.printBill();
106
107         OrderRepository repo = new OrderRepository();
108         repo.orderId = order.orderId;
109         repo.databaseName = "OrdersDB";
110         repo.saveOrder();
111
112         EmailService email = new EmailService();
113         email.senderEmail = "shop@example.com";
114         email.receiverEmail = "alice@example.com";
115         email.message = "Your order invoice";
116         email.sendInvoiceEmail();
117     }
118 }
119
```



```
C:\Windows\system32\cmd.exe - pause
Order taken
Order details printed
Total bill: 110.0
Order saved
Invoice email sent
Press any key to continue . . .
```



```
1  interface Notification {
2      void send();
3      boolean validate();
4      void logNotification();
5  }
6
7  class EmailNotification implements Notification {
8      String recipient;
9      String message;
10     String timestamp;
11
12     EmailNotification(String recipient, String message) {
13         this.recipient = recipient;
14         this.message = message;
15         this.timestamp = "2025-12-29 10:00";
16     }
17
18     public void send() {
19         if (validate()) {
20             System.out.println("Email sent to " + recipient + ": " + message);
21             logNotification();
22         } else {
23             System.out.println("Invalid email notification");
24         }
25     }
26
27     public boolean validate() {
28         return recipient.contains("@") && !message.isEmpty();
29     }
30
31     public void logNotification() {
32         System.out.println("Email logged at " + timestamp);
33     }
34 }
35
36 class SMSNotification implements Notification {
37     String recipient;
38     String message;
39     String timestamp;
40
41     SMSNotification(String recipient, String message) {
42         this.recipient = recipient;
43         this.message = message;
44         this.timestamp = "2025-12-29 10:01";
45     }
46
47     public void send() {
48         if (validate()) {
49             System.out.println("SMS sent to " + recipient + ": " + message);
50             logNotification();
51         } else {
52             System.out.println("Invalid SMS notification");
53         }
54     }
55
56     public boolean validate() {
57         return recipient.matches("\\d{10}") && !message.isEmpty();
58     }
59
60     public void logNotification() {
61         System.out.println("SMS logged at " + timestamp);
62     }
63 }
64
65 class PushNotification implements Notification {
66     String recipient;
67     String message;
68     String timestamp;
69
70     PushNotification(String recipient, String message) {
71         this.recipient = recipient;
72         this.message = message;
73         this.timestamp = "2025-12-29 10:02";
74     }
75
76     public void send() {
77         if (validate()) {
78             System.out.println("Push notification sent to " + recipient + ": " + message);
79             logNotification();
80         } else {
81             System.out.println("Invalid push notification");
82         }
83     }
84
85     public boolean validate() {
86         return !recipient.isEmpty() && !message.isEmpty();
87     }
88
89     public void logNotification() {
90         System.out.println("Push notification logged at " + timestamp);
91     }
92 }
93
94 class NotificationService {
95     void sendNotification(Notification notification) {
96         notification.send();
97     }
98 }
99
100 public class Lab8Ex7 {
101     public static void main(String[] args) {
102         NotificationService service = new NotificationService();
103
104         Notification email = new EmailNotification("alice@example.com", "Your order has shipped");
105         Notification sms = new SMSNotification("0123456789", "Your OTP is 123456");
106         Notification push = new PushNotification("user123", "You have a new message");
107
108         service.sendNotification(email);
109         service.sendNotification(sms);
110         service.sendNotification(push);
111     }
112 }
113
```



```
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 . . .
```



```
1  class Report {
2      String reportName;
3      String createdAt;
4      String format;
5
6      Report(String reportName, String createdAt, String format) {
7          this.reportName = reportName;
8          this.createdAt = createdAt;
9          this.format = format;
10     }
11 }
12
13 interface ReportGenerator {
14     void generate();
15     void export();
16     void preview();
17 }
18
19 class PDFReportGenerator implements ReportGenerator {
20     Report report;
21
22     PDFReportGenerator(Report report) {
23         this.report = report;
24     }
25
26     public void generate() {
27         System.out.println("Generating PDF: " + report.reportName);
28     }
29
30     public void export() {
31         System.out.println("Exporting PDF");
32     }
33
34     public void preview() {
35         System.out.println("Previewing PDF");
36     }
37 }
38
39 class ExcelReportGenerator implements ReportGenerator {
40     Report report;
41
42     ExcelReportGenerator(Report report) {
43         this.report = report;
44     }
45
46     public void generate() {
47         System.out.println("Generating Excel: " + report.reportName);
48     }
49
50     public void export() {
51         System.out.println("Exporting Excel");
52     }
53
54     public void preview() {
55         System.out.println("Previewing Excel");
56     }
57 }
58
59 interface ReportDelivery {
60     void send();
61 }
62
63 class EmailDelivery implements ReportDelivery {
64     Report report;
65     String recipient;
66
67     EmailDelivery(Report report, String recipient) {
68         this.report = report;
69         this.recipient = recipient;
70     }
71
72     public void send() {
73         System.out.println("Sending " + report.reportName + " to " + recipient);
74     }
75 }
76
77 public class Lab8Ex8 {
78     public static void main(String[] args) {
79         Report pdfReport = new Report("Sales Report", "2025-12-29", "PDF");
80         ReportGenerator pdfGen = new PDFReportGenerator(pdfReport);
81         ReportDelivery emailPdf = new EmailDelivery(pdfReport, "alice@example.com");
82
83         pdfGen.generate();
84         pdfGen.export();
85         pdfGen.preview();
86         emailPdf.send();
87
88         Report excelReport = new Report("Inventory Report", "2025-12-29", "Excel");
89         ReportGenerator excelGen = new ExcelReportGenerator(excelReport);
90         ReportDelivery emailExcel = new EmailDelivery(excelReport, "bob@example.com");
91
92         excelGen.generate();
93         excelGen.export();
94         excelGen.preview();
95         emailExcel.send();
96     }
97 }
98
```



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