```
import java.io.Serializable;

public class Student implements Serializable {
    private static final long serialVersionUID = 1L;

    private int id;
    private String name;
    private double gpa;

    public Student(int id, String name, double gpa) {
        this.id = id;
        this.name = name;
        this.gpa = gpa;
    }

    public void displayInfo() {
        System.out.println("Student ID: " + id);
        System.out.println("Name     : " + name);
        System.out.println("GPA      : " + gpa);
    }
}import java.io.*;

public class StudentSerializationDemo {
    public static void main(String[] args) {
        String filename = "student.ser";

        // Creating a Student object
        Student student = new Student(101, "Alice", 3.85);
```

```
// Serialization
try (FileOutputStream fos = new FileOutputStream(filename);
     ObjectOutputStream oos = new ObjectOutputStream(fos)) {

    oos.writeObject(student);
    System.out.println("Student object serialized successfully.");

} catch (FileNotFoundException e) {
    System.out.println("File not found during serialization: " + e.getMessage());
} catch (IOException e) {
    System.out.println("IOException during serialization: " + e.getMessage());
}

// Deserialization
try (FileInputStream fis = new FileInputStream(filename);
     ObjectInputStream ois = new ObjectInputStream(fis)) {

    Student deserializedStudent = (Student) ois.readObject();
    System.out.println("\nStudent object deserialized successfully:");
    deserializedStudent.displayInfo();

} catch (FileNotFoundException e) {
    System.out.println("File not found during deserialization: " + e.getMessage());
} catch (IOException e) {
    System.out.println("IOException during deserialization: " + e.getMessage());
} catch (ClassNotFoundException e) {
    System.out.println("Class not found: " + e.getMessage());
```

```
        }
    }
}
```

```
import java.util.*;
import java.util.stream.*;

class Student {
    String name;
    double marks;

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

    public String getName() {
        return name;
    }

    public double getMarks() {
        return marks;
    }
}
```

```
public class StudentFilter {
    public static void main(String[] args) {
        List<Student> students = Arrays.asList(
            new Student("Alice", 82.5),
            new Student("Bob", 68.0),
            new Student("Charlie", 91.2),
            new Student("David", 74.3),
            new Student("Eve", 78.9)
        );

        System.out.println("Students scoring above 75%, sorted by marks (high to low):");

        students.stream()
                .filter(s -> s.getMarks() > 75)
                .sorted((s1, s2) -> Double.compare(s2.getMarks(), s1.getMarks())) // descending
order
                .forEach(s -> System.out.println(s.getName() + " - " + s.getMarks() + "%"));
    }
}
```

```
import java.util.*;

class InvalidPinException extends Exception {
    public InvalidPinException(String msg) { super(msg); }
}

class InsufficientBalanceException extends Exception {
    public InsufficientBalanceException(String msg) { super(msg); }
}

class ATM {
    private static final int PIN = 1234;
    private double balance;

    public ATM(double balance) { this.balance = balance; }

    void validatePin(int pin) throws InvalidPinException {
        if (pin != PIN) throw new InvalidPinException("Invalid PIN!");
    }

    void withdraw(double amount) throws InsufficientBalanceException {
        if (amount > balance) throw new InsufficientBalanceException("Insufficient balance!");
        balance -= amount;
        System.out.println("Withdrawn: " + amount + ", Balance: " + balance);
    }

    void deposit(double amount) {
        if (amount <= 0) System.out.println("Invalid deposit!");
```

```
        else {
            balance += amount;
            System.out.println("Deposited: " + amount + ", Balance: " + balance);
        }
    }

    double getBalance() { return balance; }
}

public class ATMSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ATM atm = new ATM(3000);

        try {
            System.out.print("Enter PIN: ");
            atm.validatePin(sc.nextInt());

            while (true) {
                System.out.println("\n1. Withdraw  2. Deposit  3. Balance  4. Exit");
                switch (sc.nextInt()) {
                    case 1 -> {
                        System.out.print("Amount to withdraw: ");
                        try { atm.withdraw(sc.nextDouble()); }
                        catch (InsufficientBalanceException e) { System.out.println(e.getMessage()); }
                    }
                    case 2 -> {
                        System.out.print("Amount to deposit: ");
```

```
                atm.deposit(sc.nextDouble());
            }
            case 3 -> System.out.println("Balance: " + atm.getBalance());
            case 4 -> {
                System.out.println("Thank you! Final Balance: " + atm.getBalance());
                return;
            }
            default -> System.out.println("Invalid option!");
          }
        }
      } catch (InvalidPinException e) {
        System.out.println(e.getMessage());
      }
    }
}
```