1. **Divide by Zero Handling**
   Write a Java program to take two integers as input and perform division. Handle the `ArithmeticException` if the denominator is zero.
2. **Array Index Out of Bound**
   Create an array of 5 elements. Ask the user to input an index and print the element at that index. Handle `ArrayIndexOutOfBoundsException`.
3. **Number Format Exception**
   Read a string from the user and convert it into an integer. Handle the `NumberFormatException` if the input is not a valid number.
4. **Multiple Exception Handling**
   Write a program that takes two numbers from the user and divides them. Handle both `ArithmeticException` (division by zero) and `InputMismatchException` (if the user enters non-numeric values).
5. **Custom Exception – Age Validation**
   Create a custom exception `InvalidAgeException`. Throw this exception if a user enters age less than 18 when applying for a driving license.
6. **Nested Try-Catch**
   Implement a program where:
   - The outer `try` handles an array index error.
   - The inner `try` handles division by zero.
7. **Finally Block Example**
   Write a program to demonstrate how the `finally` block executes regardless of exceptions. Example: file closing or resource cleanup.
8. **Bank Transaction Simulation**
   - Create a class `BankAccount` with a balance field.
   - Write a method `withdraw(double amount)` that throws `InsufficientBalanceException` if the withdrawal amount is greater than the balance.
   - Demonstrate exception handling when performing transactions.
9. **File Handling with Exceptions**
   Write a program to read a text file and print its content. Handle `FileNotFoundException` and `IOException`.
10. **Student Result Processing**
    - Input marks of students in 3 subjects.
    - If any mark is negative or greater than 100, throw a custom exception `InvalidMarksException`.
    - Otherwise, calculate the average and display the result.
11. **Command-Line Argument Validation**
    Write a program that accepts two numbers from command-line arguments and prints their sum. Handle exceptions if the user:
    - Provides fewer arguments
    - Provides invalid (non-numeric) arguments
12. **Library Management System**
    - Create a class Book with fields like title, author, and isAvailable.
    - Implement a method borrowBook() which throws a custom exception BookNotAvailableException if the book is already borrowed.

- Test it by creating multiple books and simulating borrowing/returning.

13. Online Shopping Cart

- Create a Cart class that can hold items with price and quantity.
- Implement a checkout() method that:
  - Throws EmptyCartException if no items are present.
  - Throws InsufficientBalanceException if total cost exceeds the available user balance.
- Demonstrate with sample input.

14. Employee Payroll System

- Input employee salary and tax rate.
- Throw InvalidSalaryException if salary < 0.
- Throw InvalidTaxRateException if tax < 0 or tax > 100.
- Compute net salary otherwise.

15. Airline Ticket Reservation

- A class Flight has a limited number of seats.
- Implement a method bookTicket(int seats) that throws:
  - OverbookingException if requested seats exceed available seats.
- Also handle exceptions if the user provides negative or zero seats.

16. Banking with Nested Exceptions

- Create a program with deposit() and withdraw() methods.
- Use nested try-catch:
  - Outer try for invalid transaction type (e.g., deposit negative amount).
  - Inner try for insufficient balance during withdrawal.

17. File Upload Simulation

- Write a method uploadFile(String fileName, long fileSize) that:
  - Throws FileTooLargeException if file size > 100MB.
  - Throws UnsupportedFileTypeException if file type is not .jpg, .png, or .pdf.
- Demonstrate exception chaining by wrapping low-level exceptions (like IOException) inside custom ones.

18. Student Registration System

- Students must register with a unique ID and valid email.
- Throw DuplicateIDException if the ID already exists.
- Throw InvalidEmailException if email format is wrong.
- Implement a registration method that handles these gracefully.

19. Railway Reservation System with Exception Propagation

- A method reserveSeat(int seatNo) in Train class throws SeatNotAvailableException.
- Exception should propagate through multiple methods (service layer → UI layer).
- Show how the exception bubbles up and is finally caught in main().

20. Banking ATM Simulation

- ATM allows withdrawal only in multiples of 500.
- If user requests an invalid amount → throw InvalidDenominationException.
- If balance is insufficient → throw InsufficientFundsException.
- Ensure ATM always prints “Thank you for using our service” using a finally block.

21. Multithreaded Exception Handling

- Create a multithreaded Java program where one thread reads numbers from a file and another thread computes the average.

- Handle exceptions like:
    - FileNotFoundException if file doesn't exist.
    - NumberFormatException if file contains non-numeric data.
    - Show how exceptions in threads can be handled properly.

22. Hospital Patient Management

- A hospital has a limit on beds.
- Throw `NoBedAvailableException` if a new patient is admitted when hospital is full.
- Throw `InvalidPatientDataException` if patient age < 0 or name is empty.

23. University Course Registration

- Each course has a limited number of seats.
- Implement `registerStudent()` that throws:
    - `SeatFullException` when capacity is exceeded.
    - `PrerequisiteNotMetException` if student hasn't completed required courses.

24. Stock Trading Platform

- Implement a method `buyShares(String stock, int quantity, double price)` that throws:
    - `InvalidStockException` if stock symbol is not recognized.
    - `InsufficientFundsException` if account balance < total purchase price.
    - `InvalidQuantityException` if quantity ≤ 0.

25. Weather Forecast API Simulation

- Write a class `WeatherService` that fetches temperature from an API.
- Throw:
    - `NetworkFailureException` if connection fails.
    - `InvalidLocationException` if city name is not found.
- Demonstrate **exception chaining** (wrap `IOException` inside custom exceptions).

26. Voting System

- Create a method `castVote(int age, String voterID)`.
- Throw:
    - `UnderageVoterException` if age < 18.
    - `DuplicateVoteException` if the same ID votes twice.
- Ensure all voters are logged in a file (with exception handling for `IOException`).

27. Hotel Booking Application

- A hotel has a limited number of rooms.
- `bookRoom(int nights)` throws:
    - `NoRoomAvailableException` if fully booked.
    - `InvalidStayDurationException` if nights ≤ 0.
- Ensure booking details are saved to file with proper exception handling.

28. Banking Transaction Logs

- Create a system where all banking transactions are stored in a file.
- If file is missing → throw `TransactionLogNotFoundException`.
- If data is corrupted → throw `CorruptedDataException`.
- Use a `finally` block to ensure file resources are always closed.

29. Online Food Delivery System

- `placeOrder(String foodItem, int quantity)` throws:
  - `OutOfStockException` if the item is unavailable.
  - `InvalidQuantityException` if quantity ≤ 0.
- Simulate multiple restaurants, each with its own menu.

30. Car Rental System

- Cars can be rented if available.
- Throw:
  - `CarNotAvailableException` if requested car is already rented.
  - `InvalidRentalPeriodException` if rental days ≤ 0.
- Ensure that returned cars update availability correctly.

31. Banking Loan Application

- A method `applyForLoan(double amount, int creditScore)` throws:
  - `LowCreditScoreException` if creditScore < 600.
  - `InvalidLoanAmountException` if amount ≤ 0 or amount > max loan limit.

32. E-Learning Platform

- Students must enroll in courses online.
- Throw:
  - `DuplicateEnrollmentException` if the student is already enrolled.
  - `InvalidCourseException` if the course doesn't exist.
- Handle exceptions when saving student data to files.

33. Smart Home IoT System

- Devices (lights, AC, heater) can be turned on/off.
- Throw:
  - `DeviceNotConnectedException` if the device is offline.
  - `InvalidDeviceCommandException` if an unsupported action is requested.
- Demonstrate exception propagation across multiple IoT controllers.

34. Airline Luggage Handling

- Each passenger has a luggage weight limit.
- Throw:
  - `OverweightLuggageException` if weight > limit.
  - `InvalidLuggageException` if weight ≤ 0.

35. Taxi Ride-Hailing App

- A customer requests a ride.
- Throw:
  - `NoDriverAvailableException` if no drivers are free.
  - `InvalidDestinationException` if the destination is not serviceable.

36. Banking Multi-Account Transfer

- Create a method `transfer(Account from, Account to, double amount)` that throws:
  - `InsufficientFundsException` if sender doesn't have enough money.
  - `InvalidTransferAmountException` if amount ≤ 0.
- Ensure transactions are **atomic** (if exception occurs, rollback transfer).

37. Cloud Storage Service

- A user uploads files to cloud storage.
- Throw:

    - `QuotaExceededException` if user exceeds storage limit.
    - `InvalidFileNameException` if file name has forbidden characters.
- Always close streams in a `finally` block.

38. Gaming Leaderboard System

- Players score points in a game.
- Throw:
    - `InvalidScoreException` if score < 0.
    - `PlayerNotFoundException` if updating score for a non-existing player.
- Store results in a file with exception handling.

39. Online Banking with Multi-Level Exception Handling

- Create a method `performTransaction()` that internally calls `authenticateUser()`, `validateAccount()`, and `processTransaction()`.
- Each method may throw its own exception, and they should propagate up to `main()`.
- Show how exception propagation is handled.

40. Space Mission Control (Fun / Creative)

- A rocket launch simulation throws:
    - `FuelShortageException` if fuel < required level.
    - `EngineFailureException` if engine test fails.
    - `WeatherNotSuitableException` if launch conditions are unsafe.
- Show how multiple exceptions are caught and handled before launch.

41. Online Exam System

- Each student must log in with username + password.
- Throw `InvalidCredentialsException` if login fails.
- During exam:
    - If time limit exceeded → throw `TimeOutException`.
    - If student tries to access invalid question index → throw `QuestionNotFoundException`.