Java Assignments: Exception Handling without Custom Exceptions

1. Banking – ATM Withdrawal

Real-World Meaning:
A customer uses an ATM. They enter an amount to withdraw.
You must handle cases where:
- The account does not have enough money.
- The input is not a valid number.
- The input amount is zero or negative.

Requirements:
1. Create a class ATM with:
- A double balance field (e.g., 10000).
- A withdraw(double amount) method.
2. In withdraw():
- If amount > balance → throw ArithmeticException.
- If amount <= 0 → throw IllegalArgumentException.
3. In main():
- Take withdrawal amount as input using Scanner.
- Use try-catch to handle:
- NumberFormatException for invalid numbers.
- ArithmeticException for insufficient funds.
- IllegalArgumentException for zero/negative amounts.
4. Print messages like:
- “Insufficient balance. Available: …”
- “Invalid amount entered.”
- “Please enter a number.”

2. E-Commerce – Price Calculation

Real-World Meaning:
When checking out in an online store, users must enter price and quantity.
You must make sure:
- The inputs are valid numbers.
- Neither price nor quantity is zero/negative.
- No calculation overflow occurs.

Requirements:
1. Create a Checkout class with method calculateTotal(double price, int quantity).
2. In calculateTotal():
- If price <= 0 or quantity <= 0 → throw IllegalArgumentException.
- Multiply price * quantity and return total.
3. In main():
- Read inputs from Scanner as Strings.
- Convert using Double.parseDouble() and Integer.parseInt() in try block.
- Catch:
- NumberFormatException for non-numeric input.
- IllegalArgumentException for zero/negative values.

- ArithmeticException for overflow (simulate).
- Always print “Checkout process complete” in finally.

3. File Processing – Read Customer Data

Real-World Meaning:
A bank reads customer balances from a file.
The system must:
- Handle missing files.
- Handle wrongly formatted data.
- Close the file properly even if errors occur.

Requirements:
1. Have a file customers.txt with lines like:
John,5000
Alice,7000
Bob,abc
2. Create CustomerFileReader class:
- Open file with BufferedReader.
- Split each line by comma.
- Parse balance as double.
3. In try-catch:
- Catch FileNotFoundException if file missing.
- Catch NumberFormatException if balance invalid.
- Catch IOException for read issues.
4. Close file in finally or use try-with-resources.

4. Travel Booking – Seat Allocation

Real-World Meaning:
You are allocating flight seats.
The system should:
- Handle invalid seat numbers.
- Prevent double booking.
- Reject non-numeric seat inputs.

Requirements:
1. Create FlightBooking class:
- boolean[] seats = new boolean[5]; (false = available, true = booked)
- Method bookSeat(int seatNumber):
- If seatNumber out of range → throw ArrayIndexOutOfBoundsException.
- If seat already booked → throw IllegalStateException.
- Else mark booked.
2. In main():
- Ask user for seat number.
- Use try-catch:
- NumberFormatException for non-numeric.
- ArrayIndexOutOfBoundsException for invalid seat.
- IllegalStateException for already booked.
3. After booking attempt, print seat status.