Object Oriented Programming using Java

Assignment - 6

Exception Handling

1. Write a program to perform division of two number after taking both the number as input from the user. Handle all the possible exceptions (/ by zero, number format etc) and display suitable messages.
2. Write a program to store some number into an array and display them. Declare an integer array of size 5 in the main method. Ask the user for number of elements he/she wants to store in the array. If number of elements is more than the size of array, throw and handle ArrayIndexOutOfBoundsException. Define a method insertIntoArray(int a[], int size) to insert data into the array and make sure user is inserting integer data only, otherwise handle the exception and display suitable message.
3. Create a custom exception class called InvalidAgeException. Write a program that prompts the user to enter their age. If the age is less than 0 or greater than 120, throw an InvalidAgeException.
4. Write a class called Account with the following properties and methods:

   Properties: String name, int acc_no,double balance
   Methods: void deposit(int amt)
   void withdraw(int amt),
   void transfer (Account acc1, Account acc2, int amt)
   Assume that an account needs to have a minimum balance of 500. If an attempt is made to withdraw or transfer, which results in balance below 500, throw a user defined exception called MinimumBalanceException.

5. You are building a simple ATM withdrawal system. Create an account class with properties like name, balance and pin. Implement the following features:
   a. The user can check the balance.
   b. The user can withdraw an amount. If the user tries to withdraw more than their balance, throw a custom exception InsufficientFundsException. If the withdrawal amount is negative or zero, throw a custom InvalidAmountException.
   c. Before making any transaction (balance check or withdraw) the pin has be entered. If the user enters an invalid PIN, throw a custom exception InvalidPinException.
   d. If the user tries to withdraw more than the maximum allowed limit, throw a custom exception WithdrawalLimitExceededException.